Lecture 17
Link Layer

ECSE 416 – Fall 2019

McGill

The Data Link Layer

Goals:

- understand principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control: *done!*
- Instantiation & implementation of link layer technologies

The Data Link Layer

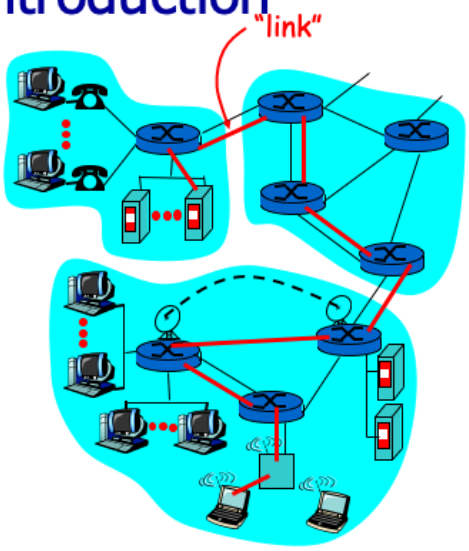
Overview:

- link layer services
- error detection & correction
- multiple access protocols and LANs
- link layer addressing & Address Resolution Protocol (ARP)
- specific link layer technologies:
  - Ethernet
  - hubs, bridges, switches
  - IEEE 802.11 LANs
  - PPP

Link Layer: Introduction

Some terminology:

- hosts and routers are nodes
- communication channels that connect adjacent nodes along communication path are links
  - wired links
  - wireless links
  - LANs
- layer-2 packet is a frame, encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to an adjacent node over a link

ECSE 416, Lecture 17

The network layer has the responsibility of transferring a datagram from one network interface to another (across the entire network).

The data-link or link layer has the responsibility of transferring a datagram over a single link – from one node (router) to an adjacent node.

At the link layer, a layer-2 packet is called a frame, and it encapsulates a datagram.

Link layer: context

- Datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- Each link protocol provides different services
  - e.g., may or may not provide reliable data transfer over link

There are many different link layer protocols (in part because we have many different transmission channels – wired, wireless, optical – and these channels have very different properties).

The services provided vary from protocol to protocol. Some provide reliable transfer, others do not.

Link Layer Services

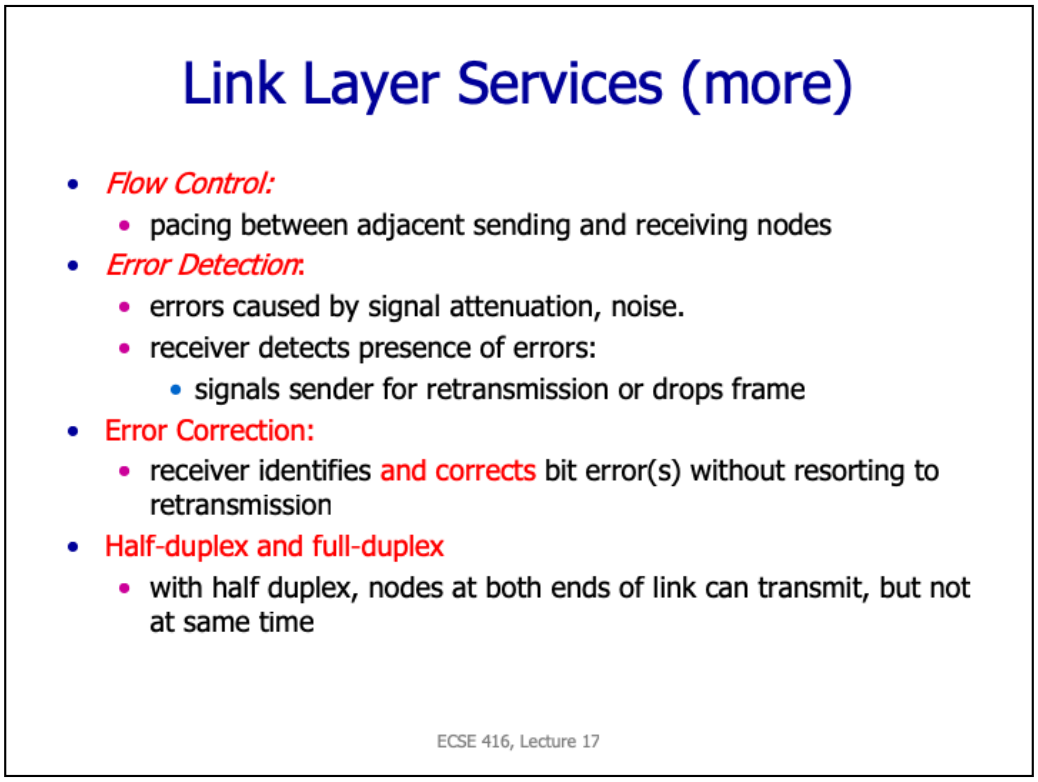
- Framing, link access:
  - encapsulate datagram into a frame, adding header & trailer
  - channel access if shared medium
  - MAC addresses used in frame headers to identify source, dest
    - different from IP address!
- Reliable delivery between adjacent nodes
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?

Some of the services include:

(1) Framing: almost all protocols encapsulate a datagram, adding header and trailer bits to form a frame.

(2) Link access: a single link is often shared between multiple communicating entities, and the link-layer protocol is responsible for providing channel access (who has the right to transmit and when).

(3) Reliable delivery: this is seldom provided in wired networks (they may do error checking, but will then just discard the datagram if they detect an error). It's more important in wireless links, which have a high loss rates (often on the order of 1 in 10^6 bits). If we have packets that are 1250 bytes (or 10,000 bits), then roughly 1 out of every 100 packets will be corrupted on a wireless channel. If we need to re-transmit all of these packets from the source host, the TCP performance will be very poor.



Other link layer services include:

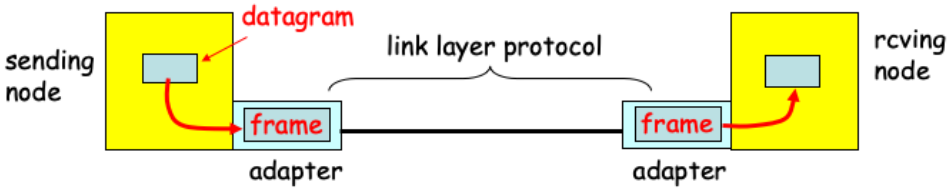
(1) Flow control: for some links we need to ensure that the sender does not overwhelm the receiver;

(2) Error detection: many link protocols include a checksum to allow for the detection of error.

(3) Error correction: some protocols even provide support for error correction, so that retransmission is unnecessary, but this is rare.

(4) Half- or Full-duplex communication: some protocols allow the nodes at both ends of a link to transmit at the same time (full-duplex); others allow only one of the nodes to transmit.

Adapters Communicating

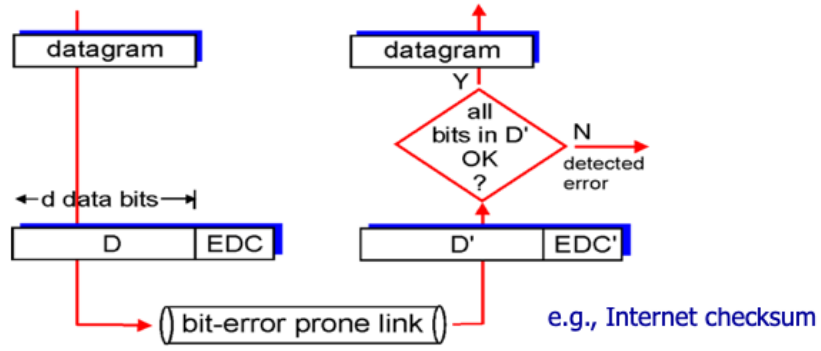


- link layer implemented in adapter (Network Interface Controller - NIC)
  - Ethernet card, PCMCI card, 802.11 card
- sending side:
  - encapsulates datagram in a frame
  - adds error checking bits, reliable data transfer, flow control, etc.
- receiving side
  - looks for errors, reliable data transfer, flow control, etc
  - extracts datagram, passes to rcving node
- semi-autonomous adapter implements link and physical layers

Error Detection

EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction

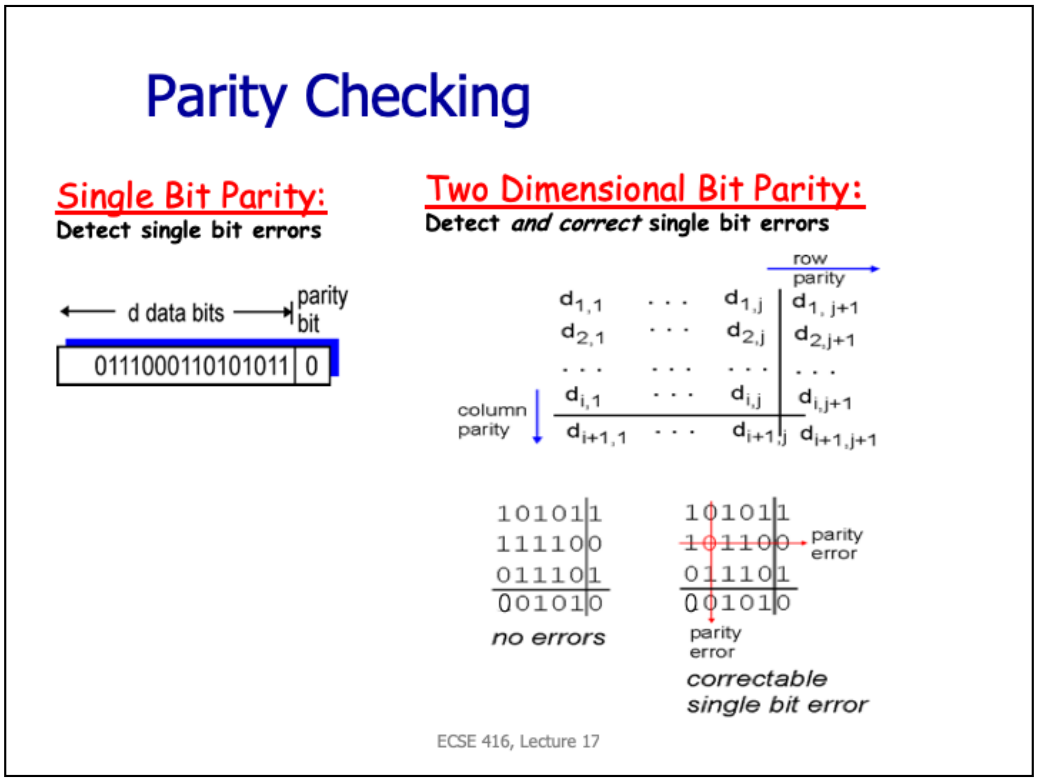


e.g., Internet checksum

ECSE 416, Lecture 17

Error detection is common in link-layer protocols; error correction is less common, but is provided in a few protocols.

These mechanisms are supported by adding error detection and correction (EDC) bits either in the header or after the data.



In single bit parity checking, we add one parity bit to the data. This allows us to detect single bit errors (or an odd number of errors).

In two-dimensional parity checking, the data is written as a matrix and a parity bit is calculated for each row and column. Single bit errors can be both detected and corrected. Two and three bit errors can be corrected, as can an odd number of errors. Some patterns of four bit errors can go undetected.

Checksumming: Cyclic Redundancy Check

- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- widely used in practice

bit pattern

$D * 2^r$ XOR R

mathematical formula

ECSE 416, Lecture 17

The cyclic redundancy check is a common approach to error checking. It is inexpensive to implement in hardware because the calculations can be executed using shifters and XORs.

To calculate the CRC bits that we use for error detection, we view the data bits D as a binary number and choose a generator G consisting of r+1 bits. We then calculate remainder bits so that <D,R> is exactly divisible by G. Note: the division is a special form of modulo-2 arithmetic. We cannot think of it like ordinary binary division.

Alternative view:

We can also interpret the data as a polynomial. i.e. 1011 = $x^3 + x + 1$.

The procedure is then to multiply the polynomial by x^r. (This is equivalent to the zero padding by r bits).

We then perform polynomial division, but use modulo-2 arithmetic when calculating the coefficients.

CRC Example

Want:

$D \cdot 2^r$ XOR R = nG

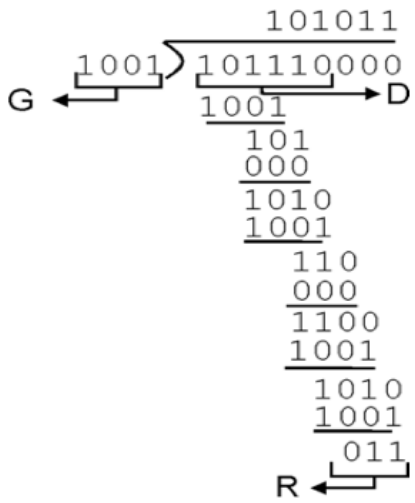
equivalently:

$D \cdot 2^r$ = nG XOR R

equivalently:

if we divide $D \cdot 2^r$ by G, want remainder R

$$R = \text{remainder}\left[\frac{D \cdot 2^r}{G}\right]$$



ECSE 416, Lecture 17

Here we see an example.

In the polynomial representation, $D = x^5 + x^3 + x^2 + x$.

$D.2^r = x^8 + x^6 + x^5 + x^4$.

We then divide by $G = x^3 + 1$ to calculate the remainder $R = x + 1$. The division proceeds as in the diagram in the slide, where the binary digits represent the polynomial coefficients.

$<D,R> = x^8 + x^6 + x^5 + x^4 + x + 1$.

If we multiply G by $x^5 + x^3 + x + 1$, we get:

$x^8 + x^5 + x^6 + x^3 + x^4 + x + x^3 + 1 = x^8 + x^6 + x^5 + x^4 + x + 1$

So <D,R> is divisible by G.

CRC Error Detection Capabilities

- CRC codes with more than one non-zero bit are capable of detecting all single bit errors.
- If x + 1 is a factor of the generator, then the CRC can detect all odd errors (this means that the generator is a multiple of 11).
- If the generator polynomial has degree r (i.e. it consists of r+1 bits), then all burst errors of length r can be detected.
- A burst error of length 5 is of the form 00011111000 (the number of 0s on either side can vary, but there must be 5 ones in a row)

ECSE 416, Lecture 17

(1) Any single bit error can be expressed in the form x^j for some j. The received bitstream can be represented as $s(x)g(x)+x^j$ for some polynomial s(x). If g(x) has more than one term, then it is clear that x^j can not be divided by g(x) without generating a remainder.

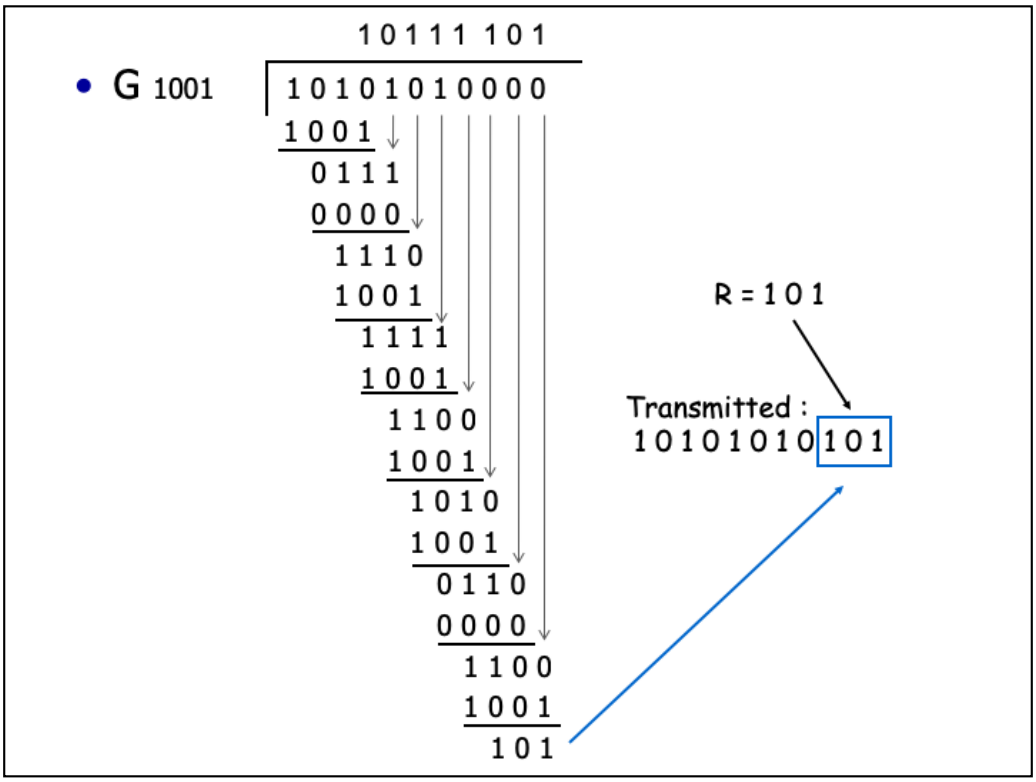
(2) If b(x) = x+1 is a factor, then g(x) = m(x)b(x) must have an even number of 1s. We can see that g(1) is equal to the sum of the polynomial coefficients of g. For the polynomial b(x) = x+1, we have b(1) = 1 + 1 = 0. Therefore g(1) = m(1)b(1) = m(1).0 = 0, so g(x) must have an even number of 1s. If we multiply g(x) by any polynomial the result retains its even parity. If there is an odd number of errors, then the received bitstream will have odd parity, and we can detect that an error must have occurred.

(3) Any burst error of length s+1 can be represented as:

$(x^s + x^{s-1} + x^{s-2} + \ldots + x + 1)x^k$ for some k. We have seen that xk is not divisible by g(x). The expression in brackets will also not be divisible if s<r. So we can detect any burst error of length r or less. If the generator is not all ones, then we can also detect a burst error of length r+1.

CRC Quiz

- Say we have data D = 10101010.
- Consider the generator G = 1001
- What is the value of R?
- What are the transmitted bits?


G 1001
10111 101
1010101 0000
1001
0111
0000
1110
1001
1111
1001
1100
1001
1010
1001
0110
0000
1100
1001
101
R = 101
Transmitted :
10101010 101

Multiple Access Links and Protocols

Two types of "links":

- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)

shared RF (satellite)

numans at a cocktail party (shared air, acoustical)

ECSE 416, Lecture 17

There are two types of links:

(1) Point-to-point (e.g. a wired connection such as a link between an Ethernet switch and a host); and

(2) Broadcast (shared wire or medium, e.g. Ethernet bus, 802.11 wireless LAN).

Multiple Access protocols

- Single shared broadcast channel
- Two or more simultaneous transmissions by nodes: interference
  - collision if node receives two or more signals at the same time

Multiple access protocol

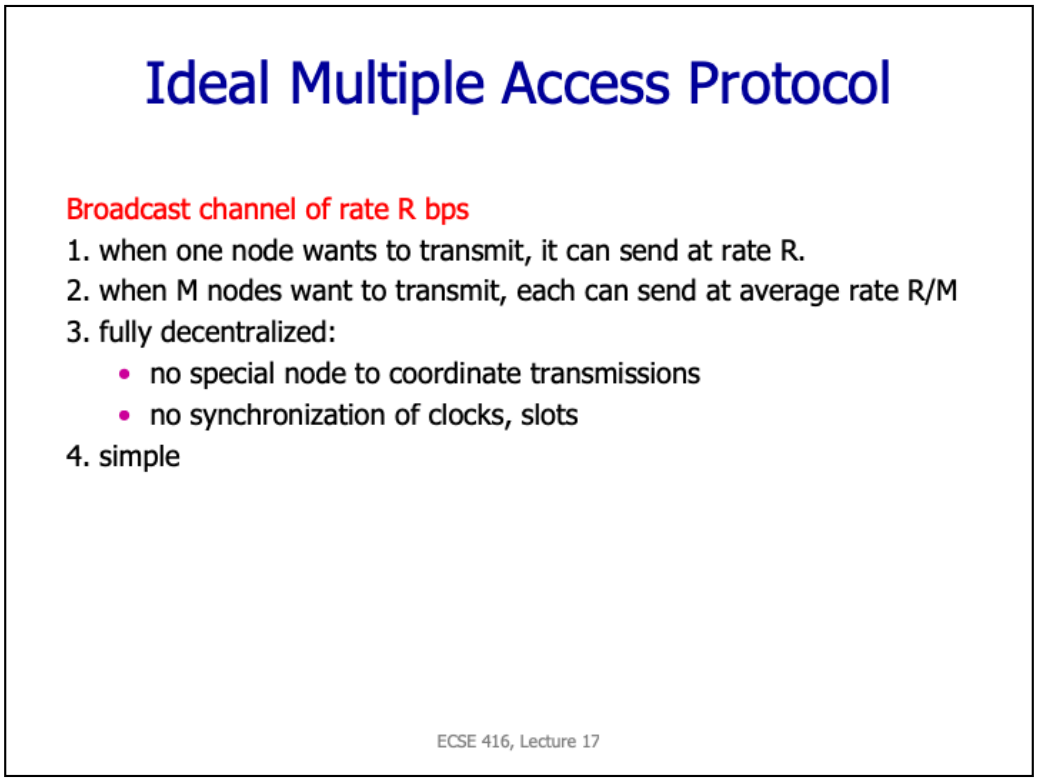
- Distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- Communication about channel sharing must use channel itself!
  - No out-of-band channel for coordination

ECSE 416, Lecture 17

When we have a shared broadcast channel, then there will be a collision if there are two or more simultaneous transmissions. Neither transmitted signal will be correctly received.

We therefore need a protocol to determine when nodes should transmit so that they can successfully share the channel.

In general, we do not have an out-of-band channel to perform coordination, so any communication about channel-sharing must be done using the transmission channel itself.

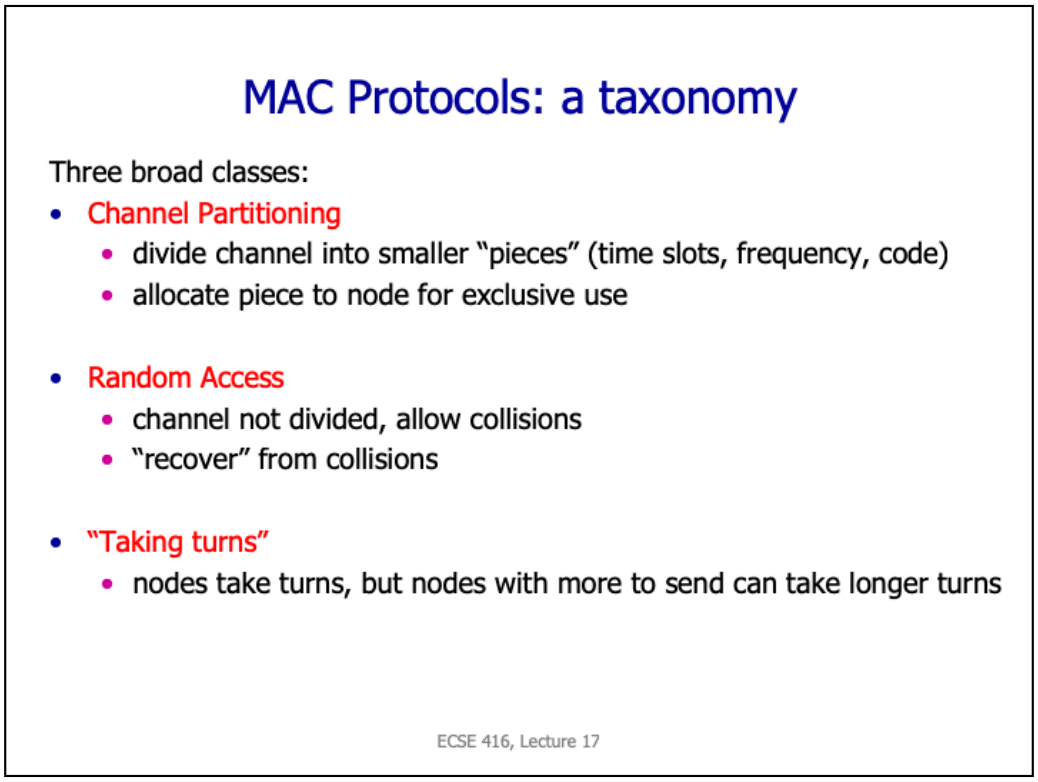


An ideal multiple access protocol has the listed characteristics.

No practical protocol can achieve all of these.

Some of the protocols work well when only one node wants to transmit, but perform poorly when many nodes want to transmit. Others work best when all nodes have data to send most of the time.

There will normally be some coordination or synchronization overhead (if not, there can be a major performance penalty).



There are three broad categories of multiple access protocols.

In channel partitioning protocols, we divide the channel into fixed pieces (time slots, frequency bands, codes) and allocate a piece to each node.

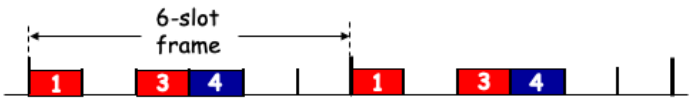
In random access protocols, we don't divide the channel. Each node transmits at the full rate of the channel when it transmits. There can be collisions, and the protocol has to specify how to recover from the collisions.

Finally there are "taking turns" protocols, that involve some sort of mechanism for performing channel reservation or providing exclusive access to a node.

Channel Partitioning MAC protocols: TDMA

TDMA: time division multiple access

- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



ECSE 416, Lecture 17

Two examples of channel partitioning protocols are time-division multiple access (TDMA) and frequency division multiple access (FDMA).

In TDMA, we divide the channel into slots. A round consists of multiple slots. In each round, we allocate one slot to each host.

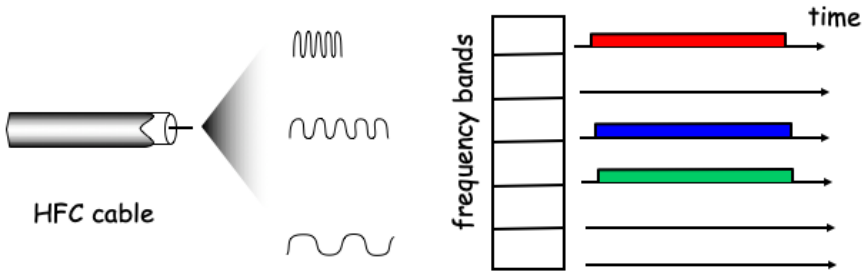
A disadvantage of this approach is that if a host has nothing to send then the slot is wasted.

If all hosts almost always have data to send (the network has high and even load) then the approach is very efficient.

Channel Partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

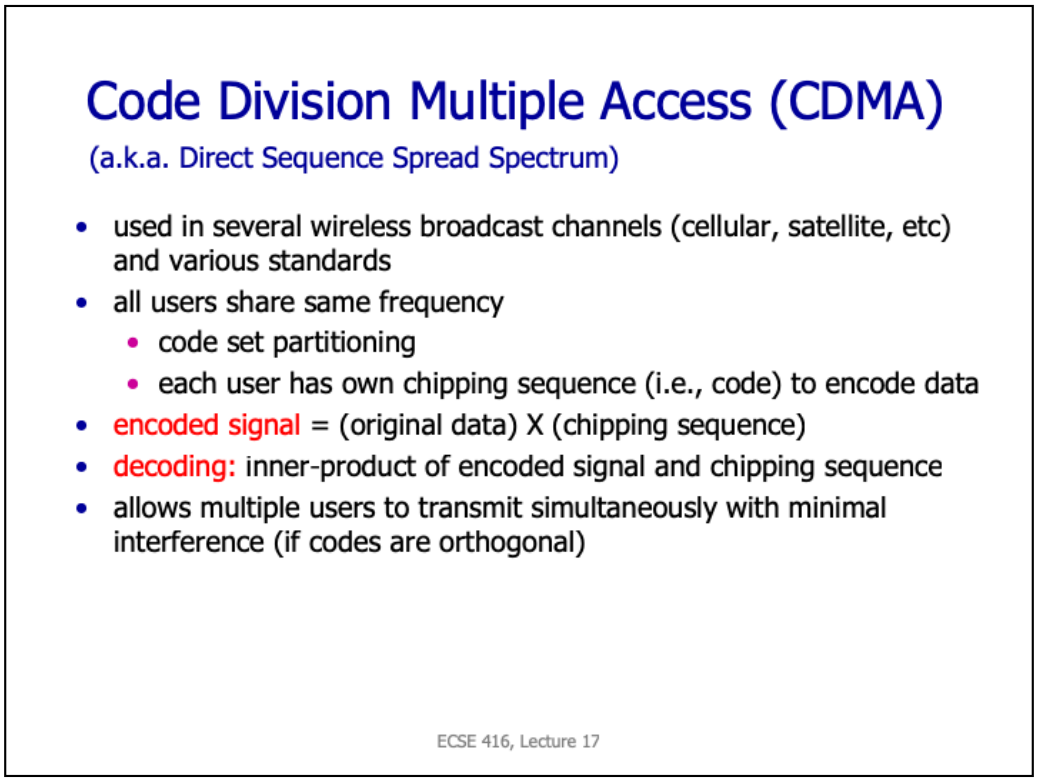
- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have packet, frequency bands 2,5,6 idle



ECSE 416, Lecture 17

FDMA divides the spectrum into frequency bands. Each user is allocated a band – since the achievable transmission rate is proportional to the bandwidth, each user can transmit at 1/M of the maximum channel rate (if there are M bands).

Again, the scheme is inefficient if multiple hosts have no data to send. If only one host has data to send, then it can only transmit at rate R/M rather than at the full rate R.



Code Division Multiple Access (CDMA) is used in numerous wireless broadcast channels.

All users share the same frequency and can transmit at the same time.

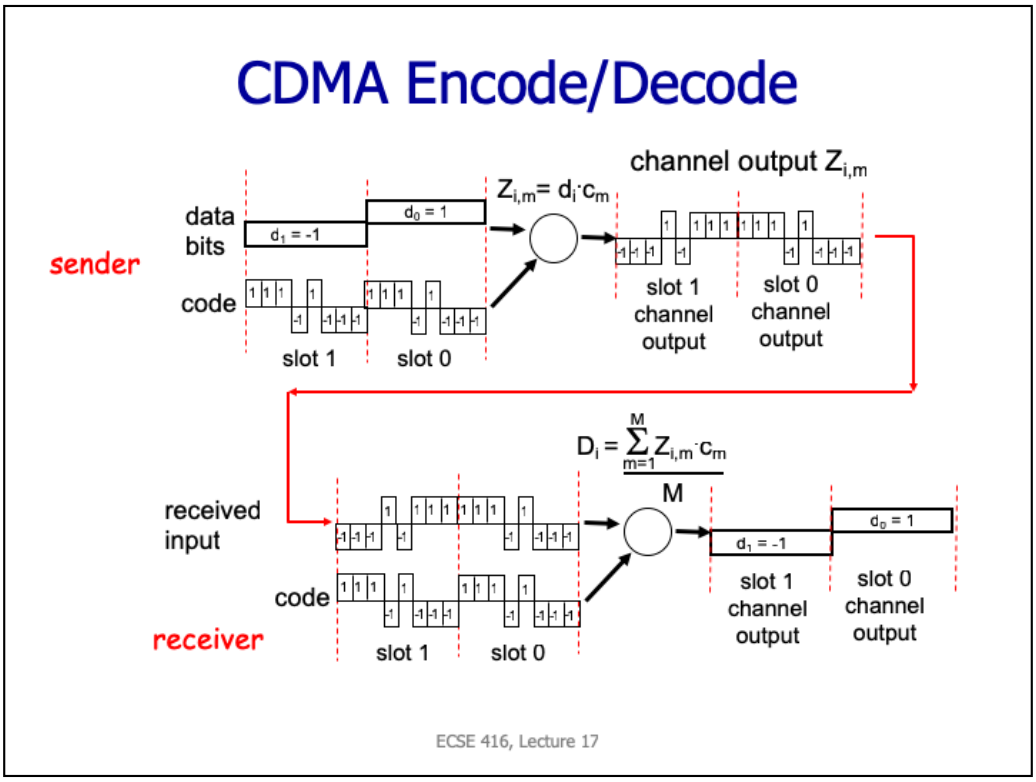
The sharing of the channel is achieved by assigning a code to each user (a chipping sequence).

The transmitted signal is the original data multiplied by the chipping sequence.

To decode, the receiver calculates the inner product of the received signal and the chipping sequence.

Multiple users can transmit simultaneously if the codes are orthogonal. Recall that orthogonal means that the inner product of the two vectors equals zero.

If there are 4 users, then we would need chipping sequences that consist of at least 4 bits in order to have 4 orthogonal codes.



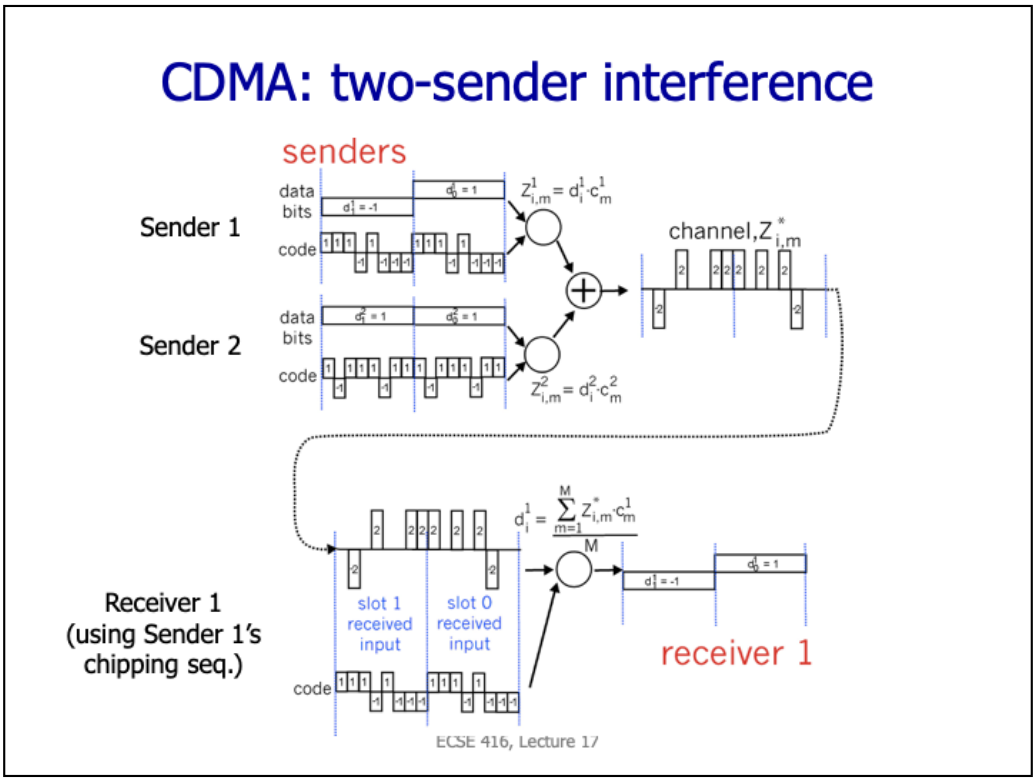
This slide depicts an example of CDMA encoding and decoding.

The original bits transmitted are [1,0], which are represented as [1,-1].

The code is [1,1,1,-1,1,-1,-1,-1].
The two encoded signals are [1,1,1,-1,1,-1,-1,-1] for the 1-bit and [-1,-1,-1,1,-1,1,1,1] for the 0-bit.

At the receiver side, we compute:
[1,1,1,-1,1,-1,-1,-1] . [1,1,1,-1,1,-1,-1,-1] / 8 = 1
and
[-1,-1,-1,1,-1,1,1,1] . [1,1,1,-1,1,-1,-1,-1] /8 = -1

to recover the transmitted data.



This slide shows how the transmitted signals of two users combine and interfere.

When we take the inner-product with respect to user 1's code, the contribution from the signal sent by user 2 is 0, because user 2's chipping sequence is orthogonal to user 1's chipping sequence.

CDMA Quiz

- Consider two CDMA codes:

A: {-1,1,1,-1,1,-1,-1,-1} and B: {-1,-1,1,1,1,-1,1,1}.

Are the codes orthogonal?

Show how sender A could transmit data = 1 and sender B data = 0 over the same time interval using these codes.

CDMA Quiz

- Consider two CDMA codes:

A: {-1,1,1,-1,1,-1,-1,-1} and B: {-1,-1,1,1,1,-1,1,1}.

Are the codes orthogonal?

Yes: <A,B> = 1-1+1-1+1+1-1-1 = 0.

Show how sender A could transmit data = 1 and sender B data = 0 over the same time interval using these codes.

Z = 1.{-1,1,1,-1,1,-1,-1,-1} + (-1). {-1,-1,1,1,1,-1,1,1}

= {0,2,0,-2,0,0,-2,-2} (combined signal)

Decode: Z.A/8 = {0,2,0,-2,0,0,-2,-2}.{-1,1,1,-1,1,-1,-1,-1}/8

= (0+2+0+2+0+0+2+2)/8 = 1.

Z.B/8 = {0,2,0,-2,0,0,-2,-2}.{-1,-1,1,1,1,-1,1,1}/8

= (0-2+0-2+0+0-2-2)/8 = -1.

Random Access Protocols

- When node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- Two or more transmitting nodes ➜ collision
- Random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- Examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

In random access protocols, there is no a priori coordination between nodes. Nodes transmit at the full channel data rate when they have a packet to send.

Since there is no coordination, collisions can occur, and the protocol must allow nodes to detect the collisions and adjust their behaviour to recover from them.

Two of the earliest random access protocols for wireless communications were ALOHA and its slotted version.

The more common protocol nowadays is Carrier-sense Multiple Access (CSMA) and it often employs ether Collision Detection (CD) or Collision Avoidance (CA).

Slotted ALOHA

Assumptions:

- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only at slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

Operation:

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with probability p until success

In slotted ALOHA, all frames are the same size and occupy one time slot.

Nodes transmit at the beginning of a slot and all nodes in the network are assumed to be synchronized.

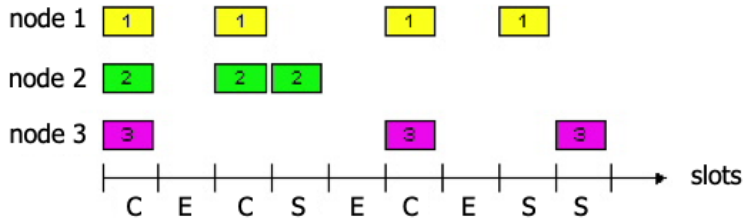
If there is a collision, we assume that all nodes can detect it. (Note: it is difficult for a wireless node to detect a collision since it generally cannot transmit and receive at the same time. In the original ALOHAnet in Hawaii in the 1970s, the network had a star topology, and the hub would send a very short acknowledgement to indicate that no collision had occurred).

When a node obtains a new frame, it transmits it in the next slot.

If there is no collision, the node can send a new frame in the next slot.

If there is a collision, it retransmits the frame in each subsequent slot with probability p until success.

Slotted ALOHA



Pros

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

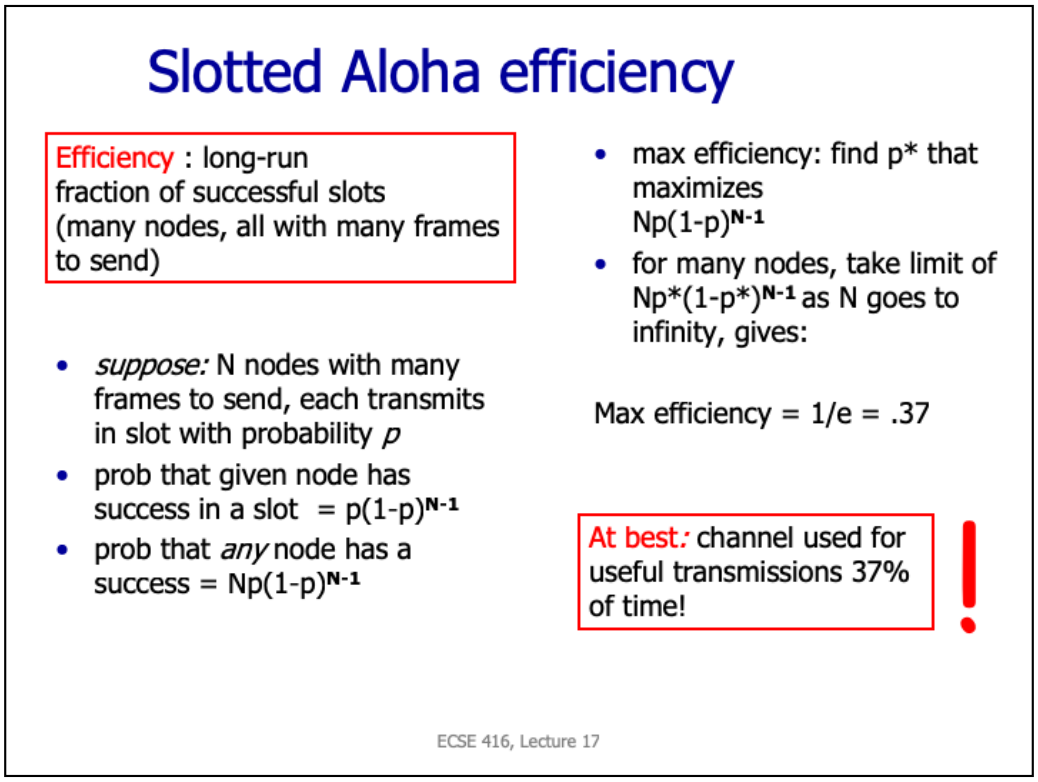
Cons

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

ECSE 416, Lecture 17

Slotted ALOHA is very efficient if a single node wants to use the channel. It is simple and decentralized. The main overhead is synchronization (and collision detection).

The main disadvantage of the protocol is the high probability of frequent collisions in a relatively heavily loaded network. The achievable efficiency is low when multiple nodes want to send most of the time.



We can evaluate the efficiency of a simplified version of slotted ALOHA.

The efficiency is the fraction of successful slots over a very long period of time. When only one node wants to transmit and always has data to send, the efficiency is close to 1.

We are interested in the efficiency when slotted ALOHA performs poorly, i.e. when many nodes want to transmit all the time.

We will analyze a slightly modified version of the protocol where N nodes always transmit in a slot with probability p (this is equivalent to the case where all nodes are in "collision" operation, which isn't a bad approximation when many nodes are trying to transmit.

The calculation of the efficiency is explained in the slide. There are three steps:

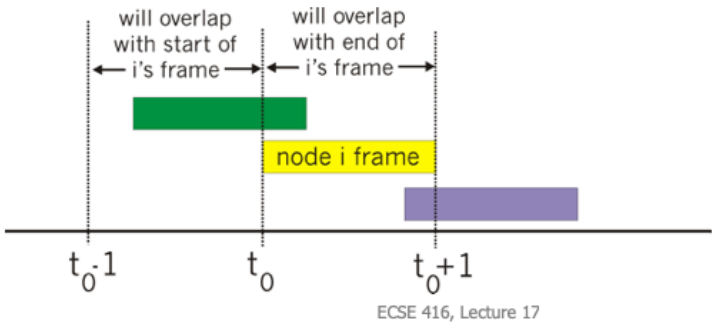
(1) Write the probability that any one node is successful in a slot;

(2) Maximize this probability (by taking the derivative and setting to zero);

(3) Use the result that $(1-1/N)^{N-1}$ goes to 1/e as N goes to infinity.

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at t_0 collides with other frames sent in $[t_0-1,t_0+1]$



ECSE 416, Lecture 17

Pure ALOHA avoids the need for synchronization by having no slots.

Unfortunately, it sacrifices efficiency in doing this.

To analyze the efficiency properly, we need to adopt a more mathematically rigorous approach, but intuitively, we can consider a single frame sent by one user and ask how often it will be successfully transmitted.

We can define an (imaginary) slot that is aligned to the frame under consideration and another slot immediately before it of the same duration. In order for there to be no collision, we need no other user to start transmitting in either of these slots (any frame transmitted by another node that overlaps the previous slot will not align exactly and will thus leak into the current slot).

We can then analyze this in a similar way as for slotted ALOHA.

Pure Aloha efficiency

P(success by given node) = P(node transmits) ·

P(no other node transmits in $[t_0-1,t_0]$ ·

P(no other node transmits in $[t_0,t_{0+1}]$

$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$

$= p \cdot (1-p)^{2(N-1)}$

... choosing optimum p and then letting n -> infty ...

$= 1/(2e) = .18$

even *worse* than slotted Aloha!

ECSE 416, Lecture 17

The probability that a node is successful is equal to the product of:

(1) The probability that it transmits;

(2) The probability that no other node transmits in the previous "slot";

(3) The probability that no other node transmits in the current "slot"

As before, we take the derivative with respect to p and set it to 0 to determine the oprimum p.

Letting n approach infinity, we see that the efficiency approaches 1/(2e) which is approximately 0.18 (half of the efficiency of slotted Aloha).

Carrier Sense Multiple Access (CSMA)

CSMA: listen before transmit:

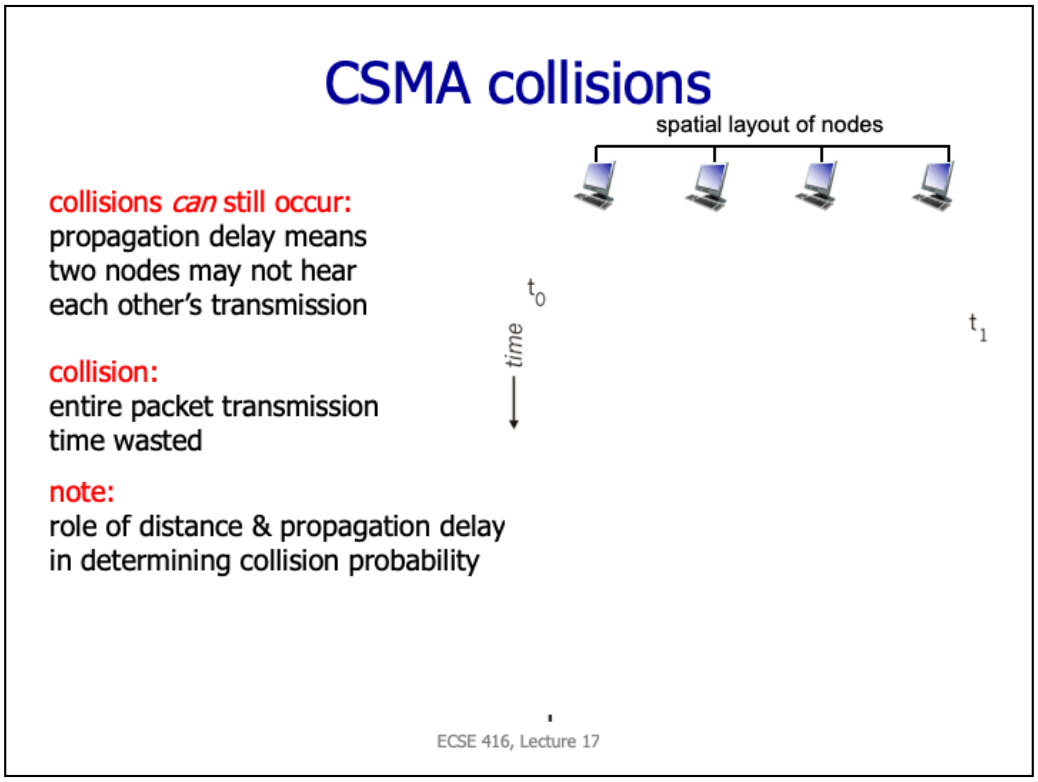
- If channel sensed idle: transmit entire frame
- If channel sensed busy, defer transmission

- Human analogy: don't interrupt others!

One way we can improve substantially on slotted and unslotted Aloha is to have nodes listen to the channel before deciding to transmit.

This allows for many of the collisions to be avoided.

If the channel is free, a node can transmit; if not, the node must wait until the channel becomes free.



Even if nodes sense the channel, collisions can still occur because there is a propagation delay.

Two nodes may decide to send before they detect each others' transmissions.

The probability of a collision depends on the propagation delay (which in turn depends on the distance between nodes and the speed of signal transmission).

In vanilla CSMA, it is assumed that nodes cannot detect collisions while they are transmitting, so an entire frame is wasted when a collision occurs.

CSMA/CD (Collision Detection)

CSMA/CD: carrier sensing, deferral as in CSMA
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

- human analogy: the polite conversationalist

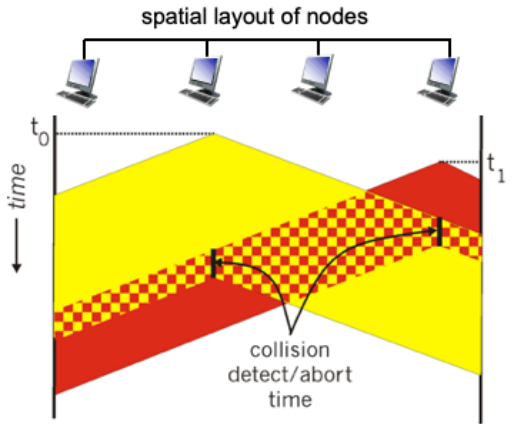
CSMA/CD adds collision detection.

We now assume that nodes can detect the state of the channel while they are transmitting. This is simple in a wired network, but not easy in a wireless network; two antennas situated in close proximity (one for sending, one for receiving and detecting collision) are likely to interfere with one another.

As soon as a collision is detected, the nodes abort their transmissions.

Usually, the nodes will continue to send an abort signal for a small time, to ensure that all nodes have enough information to detect that a collision has occurred.

CSMA/CD collision detection



CSMA/CD efficiency

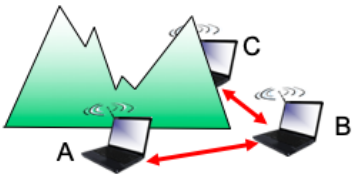
- T_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

- efficiency goes to 1
  - as t_{prop} goes to 0
  - as t_{trans} goes to infinity
- better performance than ALOHA: and simple, cheap, decentralized!

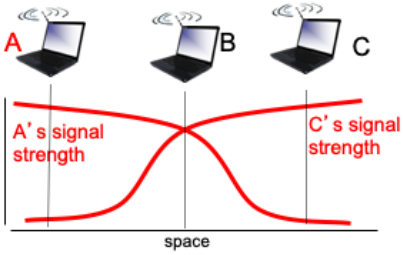
Wireless network characteristics

Multiple wireless senders and receivers create additional problems (beyond multiple access):



Hidden terminal problem

- B, A hear each other
- B, C hear each other
- A, C can not hear each other means A, C unaware of their interference at B

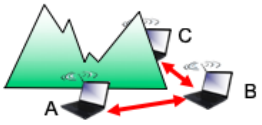


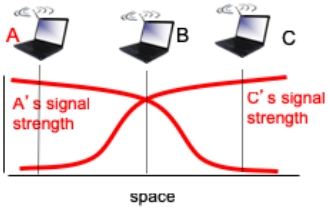
Signal attenuation:

- B, A hear each other
- B, C hear each other
- A, C can not hear each other interfering at B

IEEE 802.11: multiple access

- avoid collisions: 2^+ nodes transmitting at same time
- 802.11: CSMA - sense before transmitting
  - don't collide with ongoing transmission by other node
- 802.11: *no* collision detection!
  - difficult to receive (sense collisions) when transmitting due to weak received signals (fading)
  - can't sense all collisions in any case: hidden terminal, fading
  - goal: *avoid collisions:* CSMA/C(ollision)A(voidance)

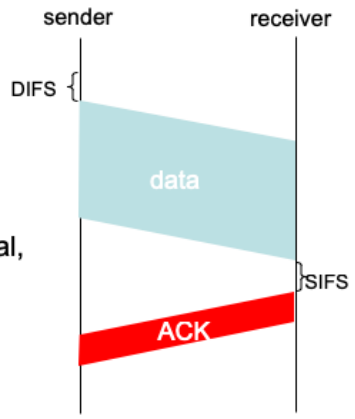




IEEE 802.11 MAC Protocol: CSMA/CA

802.11 sender

1 if sense channel idle for **DIFS** then
  transmit entire frame (no CD)
2 if sense channel busy then
  start random backoff time
  timer counts down while channel idle
  transmit when timer expires
  if no ACK, increase random backoff interval, repeat 2



802.11 receiver

- if frame received OK
  return ACK after **SIFS** (ACK needed due to hidden terminal problem)

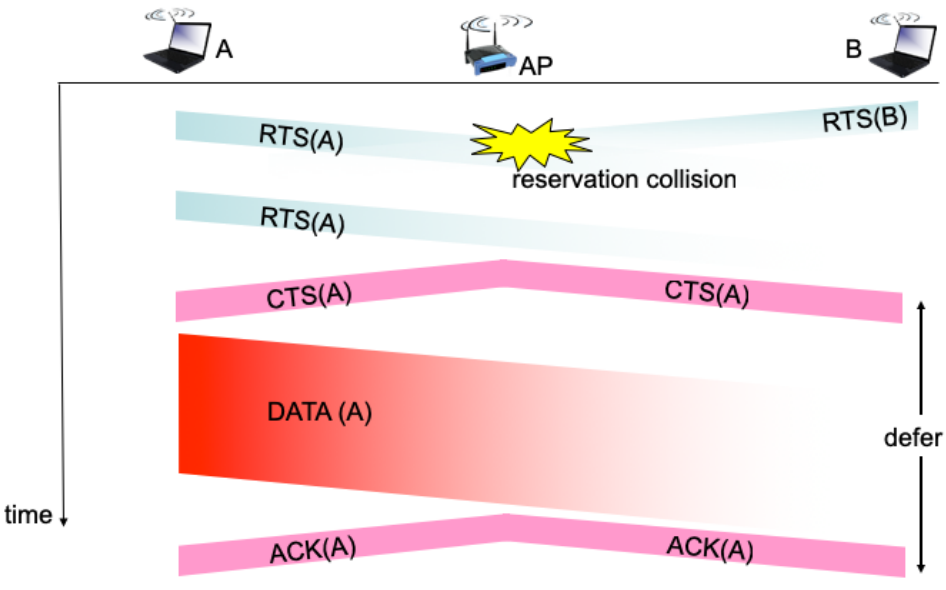
Avoiding collisions (more)

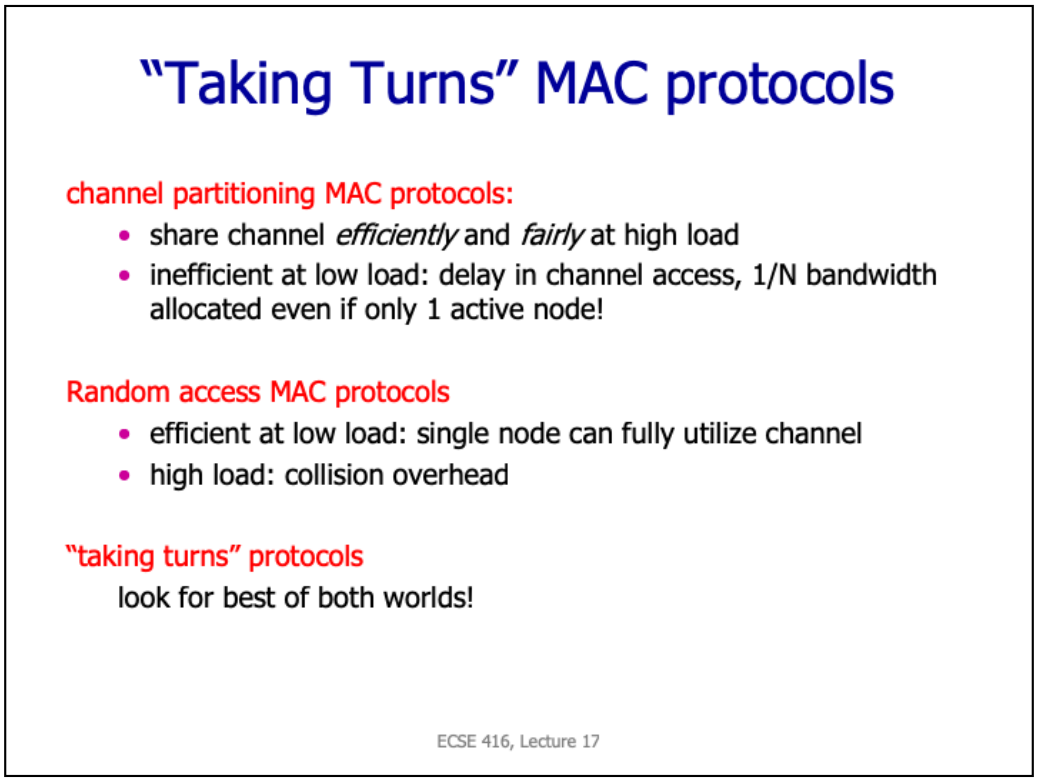
idea: allow sender to "reserve" channel rather than random access of data frames: avoid collisions of long data frames

- sender first transmits *small* request-to-send (RTS) packets to BS using CSMA
  - RTSs may still collide with each other (but they're short)
- BS broadcasts clear-to-send CTS in response to RTS
- CTS heard by all nodes
  - sender transmits data frame
  - other stations defer transmissions

avoid data frame collisions completely using small reservation packets!

Collision Avoidance: RTS-CTS exchange





The third category of MAC protocol we identified is the "taking turns" group.

This isn't a formal category, but there are protocols that have a key common property. They provide a mechanism for a node to reserve the channel for its own sole usage for a period of time.

We have seen that channel partitioning protocols are efficient when many users have data to send most of the time (heavy, constant load); random access protocols are efficient in lower load scenarios or when traffic is bursty or when a single user has a lot of data to send.

Channel partitioning protocols strive to be reasonably efficient in both high load and low load scenarios (and to be relatively insensitive to bursty traffic).

They are used in several protocols:

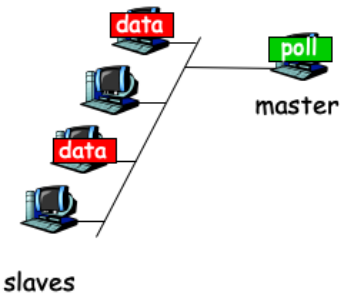
(1) Wimax: nodes are separated by long distances, so random access protocols will experience many collisions.

(2) Sensor network protocols: nodes have small energy budgets, so it is important to avoid collision as much as possible.

"Taking Turns" MAC protocols

Polling:

- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



ECSE 416, Lecture 17

One example of a "taking turns" protocol is polling.

In polling, there is a master node that invites slave nodes to transmit in turn.

The overhead here is polling. The maximum efficiency is achieved when all nodes transmit their maximum allowed data in a round. We then have a maximum efficiency of T_data / (T_data + T_polling) or equivalently 1/ (1 + T_polling/T_data). Here T_data is the maximum time allocated to a single node to send data, and T_polling is the time taken to poll a single node. In generaitng this expression, we assume that the total polling time scales linearly with the number of nodes.

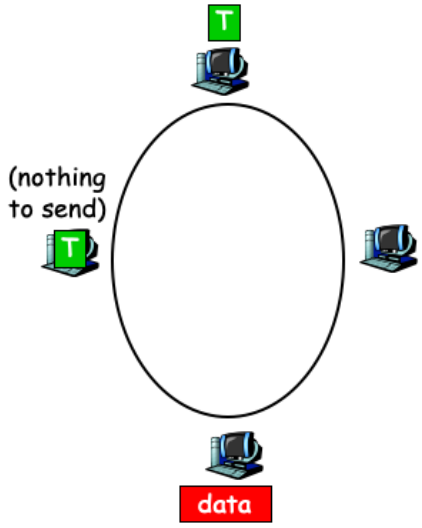
In addition to reducing the efficiency, the polling procedure introduces latency. For efficiency, we would like to make T_data large, but then a node has to wait a long time for all other nodes to do their transmissions.

Finally, the process is not particularly robust, since there is a single point of failure; if the master node fails, the protocol fails.

"Taking Turns" MAC protocols

Token passing:

- control token passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



ECSE 416, Lecture 17

An alternative to polling is token passing.

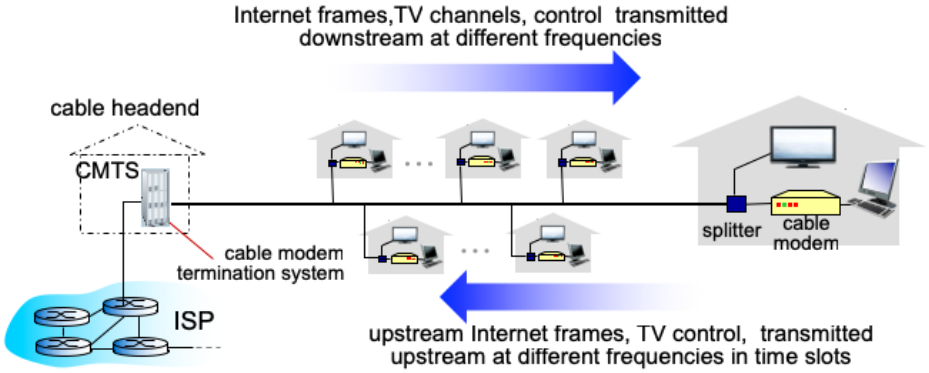
In token passing, there is a token that is passed between nodes.

A node can only transmit data when it has the token. It can only keep the token for a maximum period of time.

We have eliminated the need for a master node, but we still have a single point of failure. If the token is lost, then the protocol breaks. Practical token ring protocols include mechanisms for regenerating tokens when they are lost or corrupted.

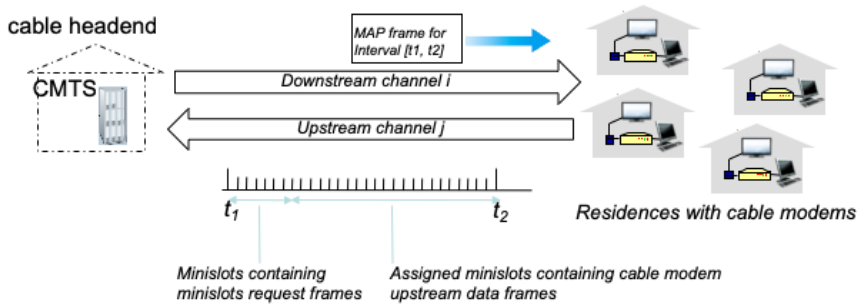
The trade-off between latency and efficiency remains a concern.

Cable access network



- *multiple* 40Mbps downstream (broadcast) channels
  - single CMTS transmits into channels
- *multiple* 30 Mbps upstream channels
  - *multiple access: all* users contend for certain upstream channel time slots (others assigned)

Cable access network

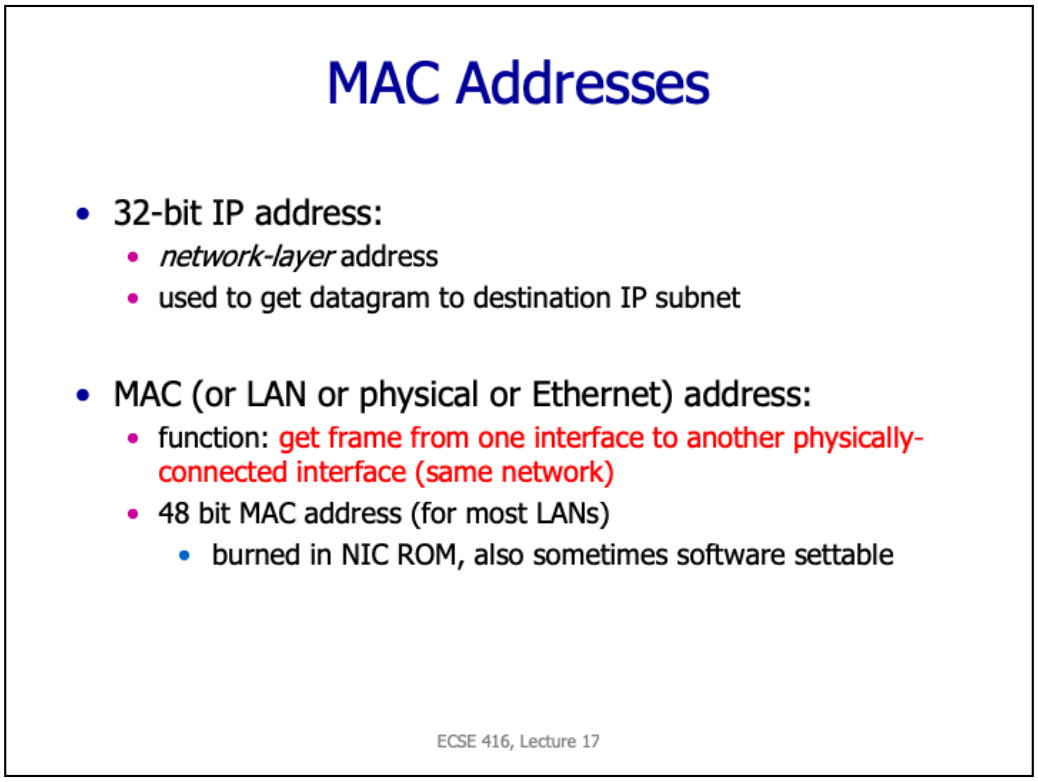


DOCSIS: data over cable service interface spec

- FDM over upstream, downstream frequency channels
- TDM upstream: some slots assigned, some have contention
  - downstream MAP frame: assigns upstream slots
  - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

Summary of MAC protocols

- *channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing
  - Bluetooth, FDDI, token ring, WiMax, sensor networks



We saw that each network interface had an IP address associated with it at the network layer.

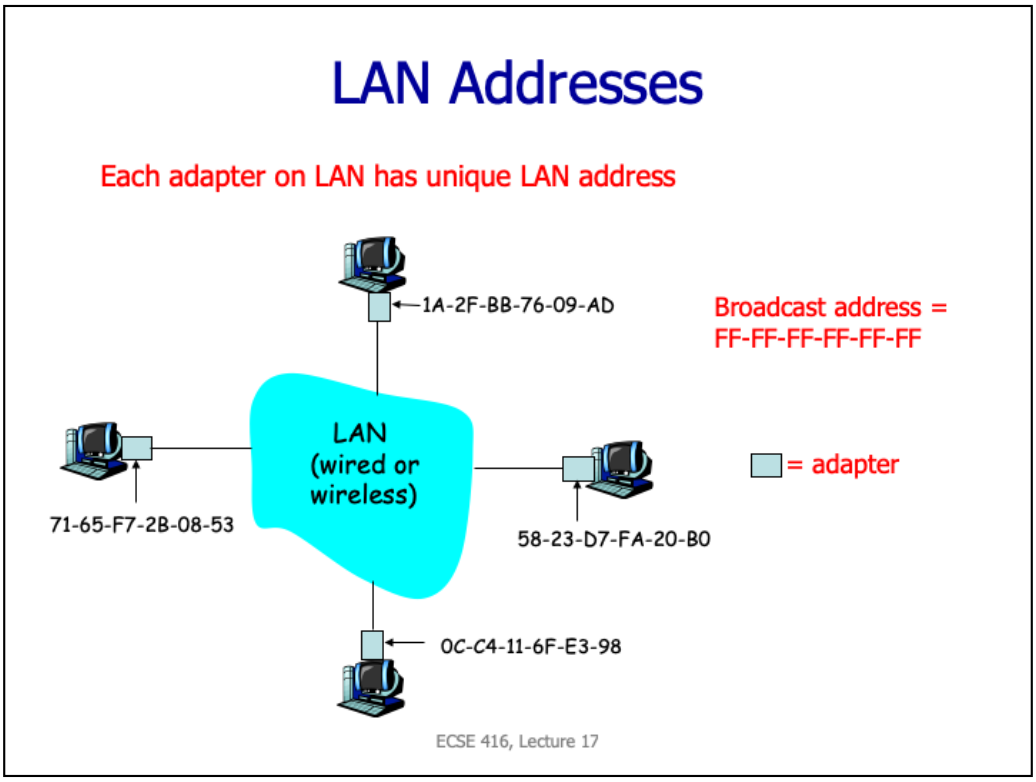
At the link layer (layer 2), there is another address – the MAC address.

The MAC address is used to get a frame from one interface to another when they are connected to the same network (they are physically connected and there is no router between them).

In most cases, the MAC address is 48 bits. It is usually burned into the ROM of the network interface card, although in some cases it can be settable in software.

One of the main reasons that we have both MAC addresses and IP addresses is that MAC addresses aren't structured hierarchically, so it is very difficult to implement efficient routing.

IP addresses play a role like postal addresses (we route to a country and then to a postcode). Routers store forwarding tables that contain prefixes of subnets; the subnets are then responsible for locating individual hosts. Prefixes of MAC addresses wouldn't make sense.



Each adapter has a unique 48 bit MAC or LAN address.

The broadcast address is all ones (i.e. FF-FF-FF-FF-FF-FF).

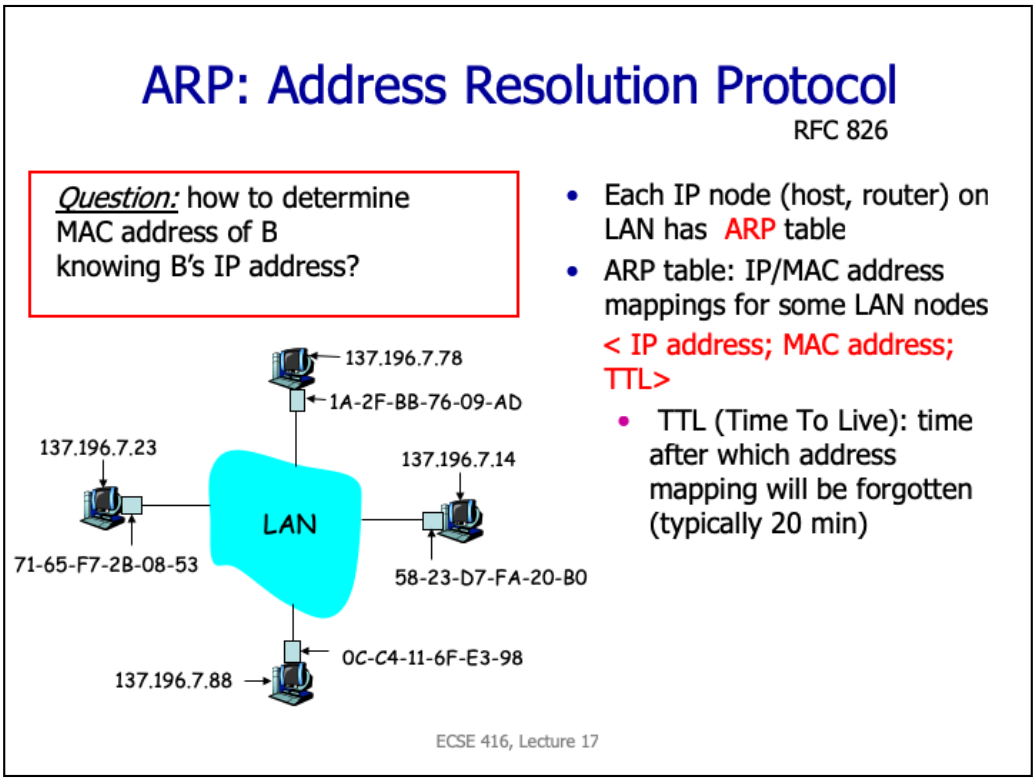
LAN Address (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - (a) MAC address: like Social Insurance Number
  - (b) IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - address depends on IP subnet to which node is attached

MAC addresses are allocated by the IEEE (hardware manufacturers buy a portion of the address space).

The flat nature of the MAC address helps portability – if I move a card from one local area network to another, I only have to inform the new LAN of the MAC address (no router outside the LAN has any knowledge of the MAC address.

The IP address is not portable, because of its hierarchical nature, but it is the hierarchical nature that makes routing datagrams across the Internet feasible.



The Address Resolution Protocol (ARP) provides a mechanism for mapping an IP address to a MAC address.

Each IP node (a host or a router) on a local area network has an ARP table that contains address mappings for some of the nodes on the LAN.

Each entry in the table takes the form <IP address; MAC address; TTL> where TTL is the time-to-live. The ARP table is soft-state, in the sense that its entries are removed and the mapping forgotten when the TTL elapses.

When nodes are active, the ARP table is frequently refreshed, so this removal has little impact.

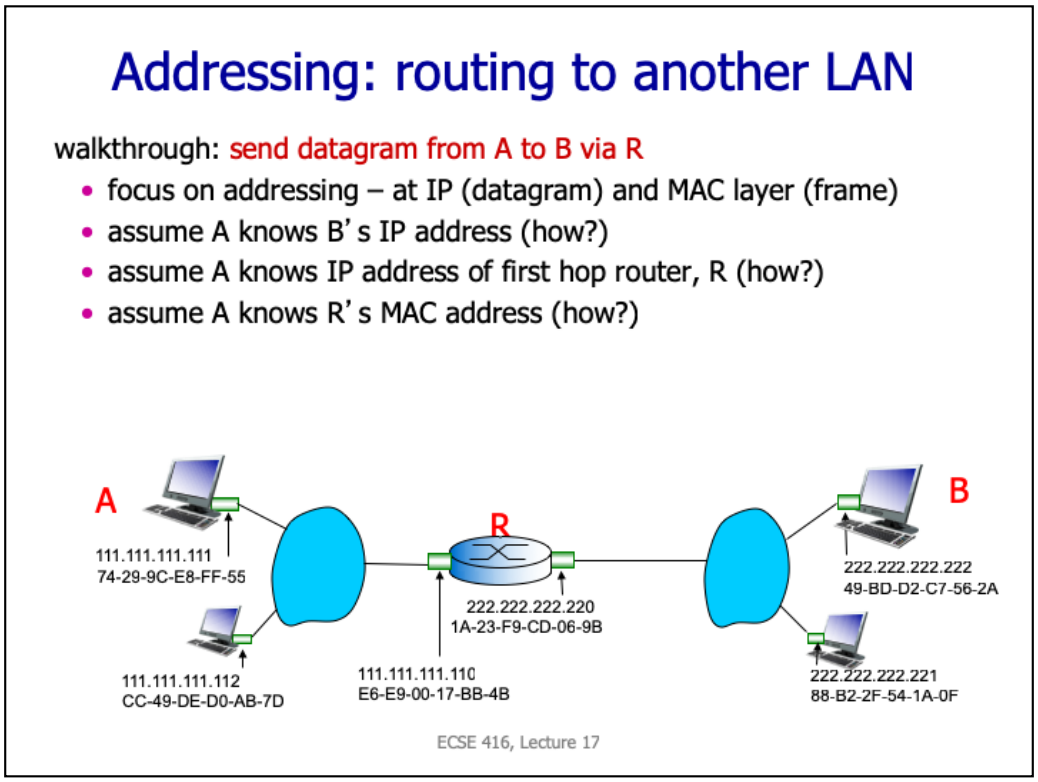
ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables without intervention from network administrator

This slide explains how A learns the MAC address of B, another node on its LAN, so that it can forward a frame to it.

A broadcasts a query message and then processes B's response.

ARP is low maintenance ("plug-and-play"); there is no need for configuration by an administrator, because the device will learn the mappings it needs as it communicates.



Routing to another LAN is similar, but this time A needs to discover the address of the interface of the outgoing router R.

Suppose A only knows B's host name.

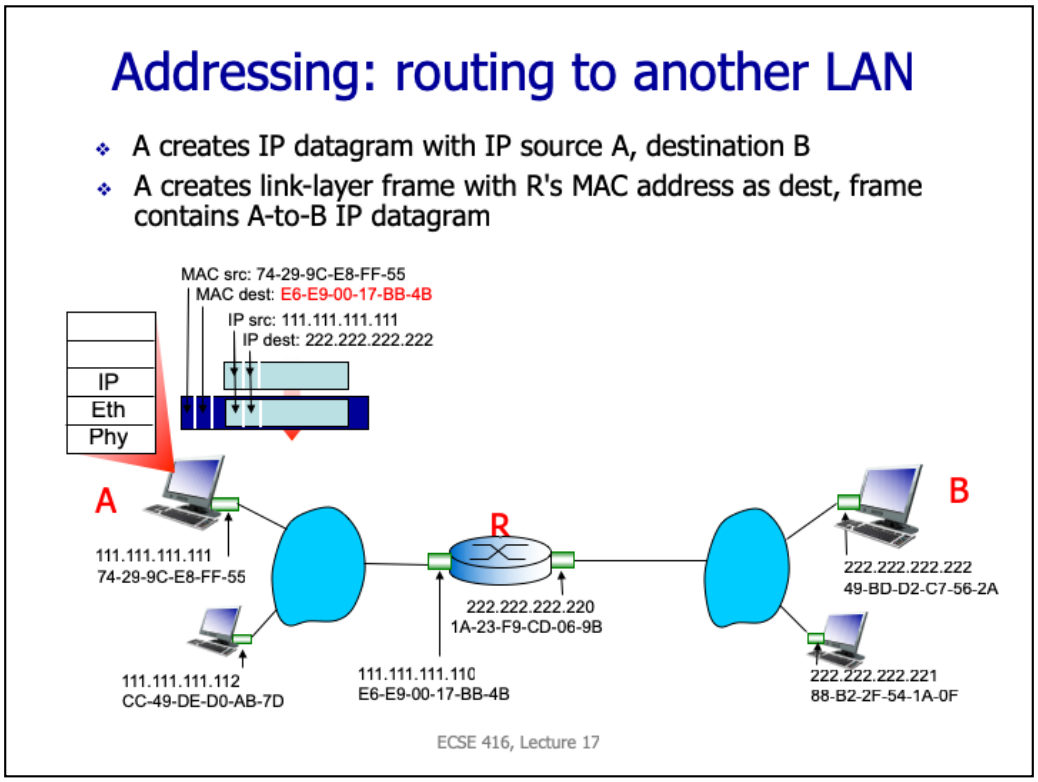
Then it will:

(1) use DNS to discover B's IP address.

(2) Look up its forwarding table to learn where to direct packets that match B's IP address

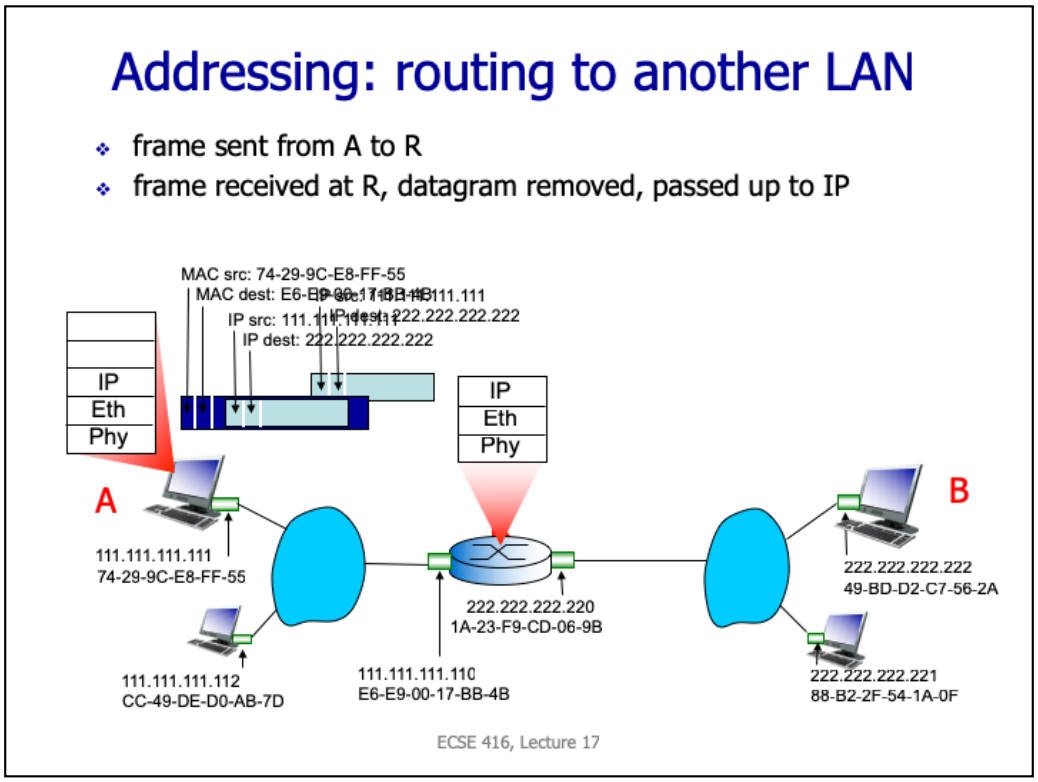
(3) In most cases, since A is a host, not a router, it is likely that it will not know any routing information for addresses outside its LAN, and it will follow the procedure of forwarding any packet destined for an unresolved IP address to its default gateway (R).

(4) If A doesn't know R's MAC address, it can send an ARP query to fill in its table – R will respond.



Once it has the addresses, A constructs a datagram with A's IP as the source and B's IP as the destination.

Then it embeds the datagram in a layer 2 frame with R's MAC address as the destination.

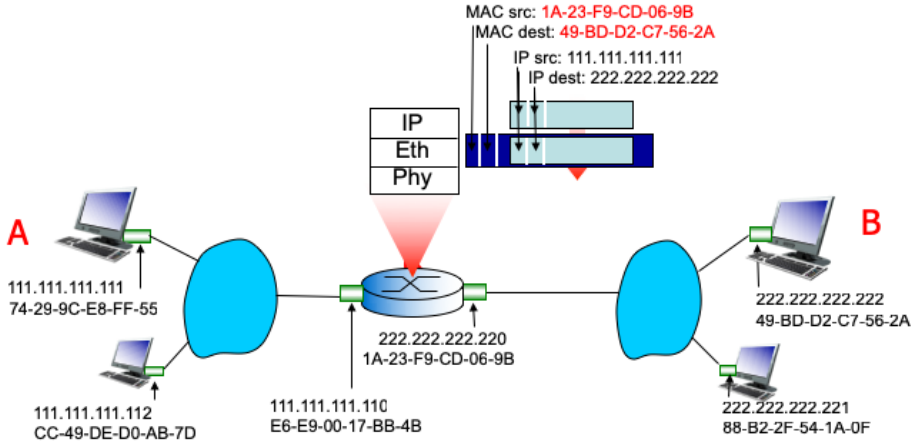


The frame is sent from A to R. R receives it, removes the datagram and processes it at the IP layer.

At the IP layer, it determines from its forwarding table the appropriate outgoing interface.

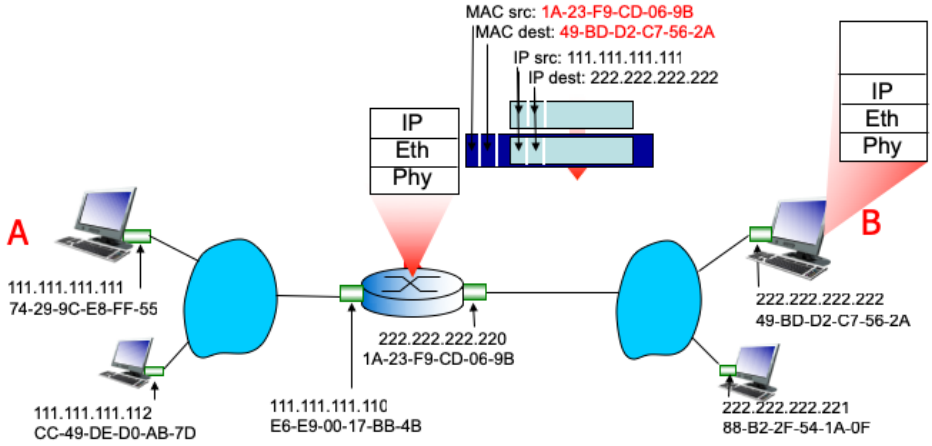
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



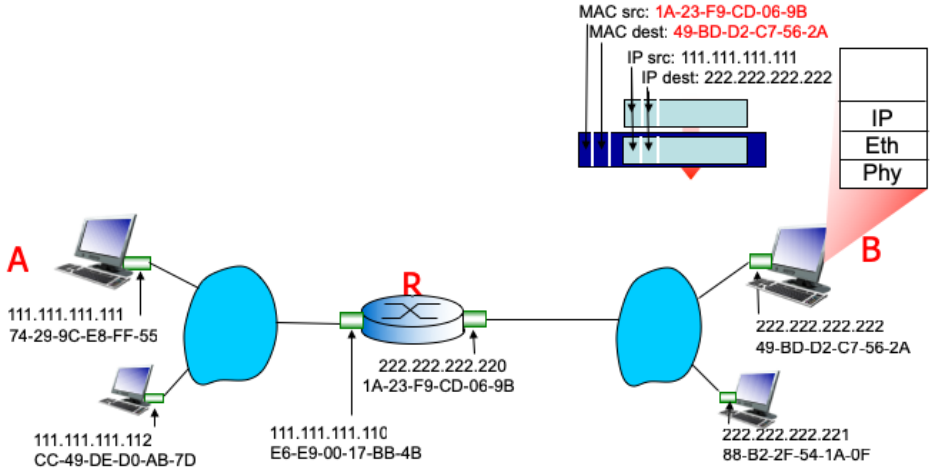
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

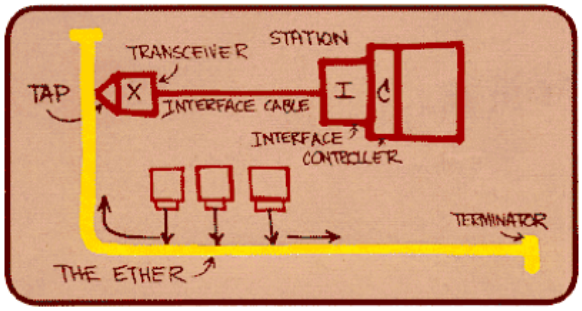
- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Ethernet

"dominant" wired LAN technology:

- cheap, $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



Metcalfe's Ethernet sketch

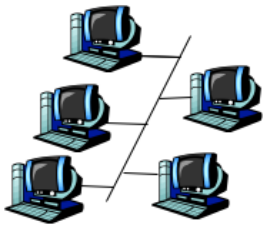
ECSE 416, Lecture 17

Ethernet is the most prevalent wired technology for local area networks.

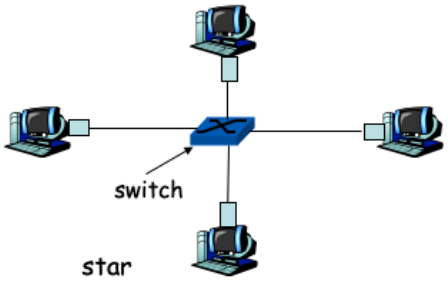
It was the first widely used technology, and although there have been many competing technologies since its development, engineers have found ways to increase the speed that can be supported (from 10 Mbps, to 100 Mbps, to 1 Gbps, to 10 Gbps, etc.) so that it is always competitive, despite being simple and cheap.

Bus and Star Topology

- bus topology popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- today: star topology prevails
  - active switch in center
  - each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)



bus: coaxial cable



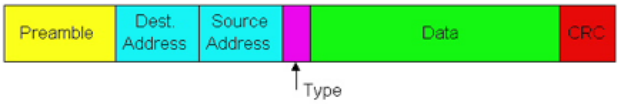
star

The bus topology was popular throughout the 90s, but switches with many ports have become relatively inexpensive, so the star topology dominates now.

In the star, each host is connected to a separate port, so the frames generated by different hosts cannot collide.

Ethernet Frame Structure

Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame



Preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

An Ethernet frame consists of:

(1) a preamble (7 bytes) that is used for clock synchronization;

(2) The destination MAC address;

(3) The source MAC address;

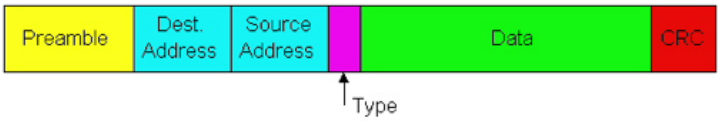
(4) The "Type" – indicates the layer 3 protocol;

(5) The Data (usually an IP datagram); and

(6) a CRC checksum (4 bytes, with a predefined, standardized generator).

Ethernet Frame Structure (more)

- Addresses: 6 bytes
  - if adapter receives frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- Type: indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- CRC: checked at receiver, if error is detected, frame is dropped (4Bytes, predefined generator)



Ethernet: Unreliable, connectionless

- connectionless: No handshaking between sending and receiving NICs
- unreliable: receiving NIC doesn't send acks or nacks to sending NIC
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP
  - otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted CSMA/CD with binary backoff

ECSE 416, Lecture 17

Ethernet provides a connectionless service (there is no handshaking between the sender and receiver) and the service is unreliable.

There are no acknowledgements and no retransmission.

If there is a corrupted frame, detected by the CRC, it is simply discarded.

Ethernet uses CSMA/CD as its MAC protocol.

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!

4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters exponential backoff: after *m*-th collision, NIC chooses *K* at random from {0,1,2,…,2^m-1}. NIC waits K·512 bit times, returns to Step 2

ECSE 416, Lecture 17

The CSMA/CD proceeds as we described the generic protocol.

When it has a frame to transmit, a node senses the channel, waiting a specified period of time to ensure that it is idle, and then sends the frame.

During transmission it continues to listen to the channel; if the bit sequence it "hears" does not match what it is sending, then it is clear that a collision has occurred. When it detects this, it sends a jam signal (of specified length in bits) to ensure that all other nodes can detect the collision.

It then enters "exponential backoff", the collision-handling mechanism. It chooses a K at random from {0,1} and waits for K.512 bit times before attempting to transmit again. If another collision occurs, it chooses K from {0,1,2,3}. For every subsequent collision, the range of possible K values doubles.

Ethernet's CSMA/CD (more)

Jam Signal: make sure all other transmitters are aware of collision; 48 bits
Bit time: .1 microsec for 10 Mbps Ethernet;
K=1023: wait time approx. 50 msec

Exponential Backoff:

- *Goal*: adapt retransmission attempts to estimated current load
  - heavy load: random wait will be longer
- first collision: choose K from {0,1}; delay is K·512 bit transmission times
- after second collision: choose K from {0,1,2,3}…
- after ten collisions, choose K from {0,1,2,3,4,…,1023}

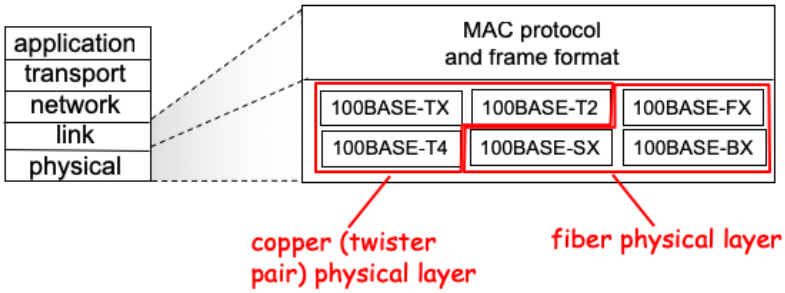
ECSE 416, Lecture 17

The goal of exponential back-off is to adapt the probability of a node attempting to retransmit so that it matches the current load on the network.

If many nodes are trying to transmit, then nodes will have a good chance of choosing large K values, so transmission attempts will be well-spaced and have a better chance of success.

802.3 Ethernet Standards: Link & Physical Layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
  - different physical layer media: fiber, cable



ECSE 416, Lecture 17

There are many different Ethernet standards (because there are many different underlying physical transmission media).

The MAC protocol and the frame format are common, but other aspects of the protocol are specific to the transmission rate and the physical medium.

Switch

- link-layer device: takes *active* role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- transparent
  - hosts are unaware of presence of switches
- plug-and-play, self-learning
  - switches do not need to be configured

Switches are layer 2 devices and are much more capable and active than hubs.

A switch stores and forwards incoming Ethernet frames. It examines the MAC address and selectively forwards it to one (or more) outgoing links.

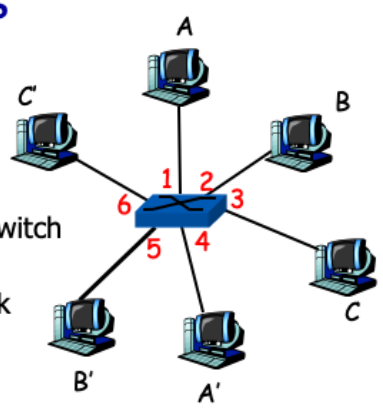
It uses CSMA/CD to access each network segment to which it is connected.

From the point of view of a host, switches are transparent – the ports of the switch do not have MAC addresses.

The switches are self-learning; they learn which ports are associated with with MAC addresses by processing incoming packets.

Switch: allows *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link
  - no collisions; full duplex
  - each link is its own collision domain
- switching: A-to-A′ and B-to-B′ simultaneously, without collisions
  - not possible with hub



switch with six interfaces (1,2,3,4,5,6)

ECSE 416, Lecture 17

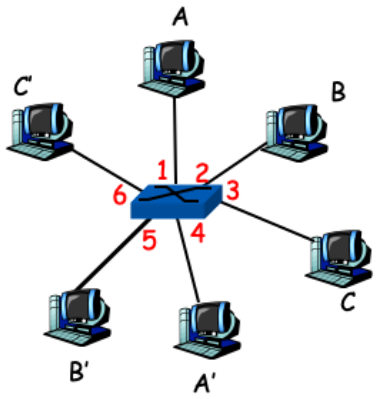
A switch separates the collision domains. Each link is its own collision domain; generally the links are full duplex, so there are effectively no collisions.

Since the switch does actual switching, as opposed to the repeating performed by a hub, it is possible for two frames to be switched simultaneously if they have different source and destination ports.

The switch does buffering, so it can introduce some queuing delay if there is a heavy load on the network.

Switch Table

- How does switch know that A′ is reachable via interface 4 & B′ is reachable via interface 5?
- Each switch has a switch table
  - Entry: (MAC address of host, interface to reach host, time stamp)
- Looks like a routing table!
- How are entries created & maintained in a switch table?
  - something like a routing protocol?



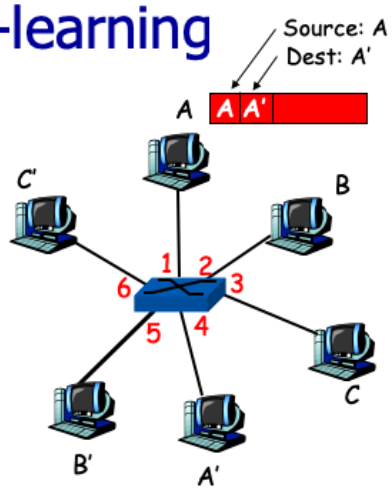
switch with six interfaces (1,2,3,4,5,6)

ECSE 416, Lecture 17

The switch maintains a switch table (similar to a forwarding table) that stores a mapping from MAC addresses to outgoing interfaces (switch ports).

Switch: self-learning

- switch learns which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	1	60

Switch table (initially empty)

The switch learns the entries in the switch table. When it receives a frame, it checks the source MAC address and if it has no entry for that address, it adds the (MAC address, incoming interface) pair to its switch table.

Switch: frame filtering/forwarding

When frame received:

1. record link associated with sending host
2. index switch table using MAC dest address
3. if entry found for destination
   then {
     if dest on segment from which frame arrived
       then drop the frame
       else forward the frame on interface indicated
   }
   else flood

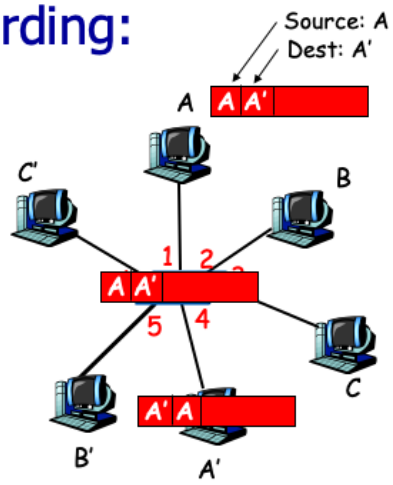
forward on all but the interface on which the frame arrived

ECSE 416, Lecture 17

If a switch doesn't have an entry for a destination MAC address in its switch table then it forwards the frame on all output interfaces, except for the one on which it was received.

Self-learning & forwarding: example

- frame destination unknown: flood
- destination A location known: selective send



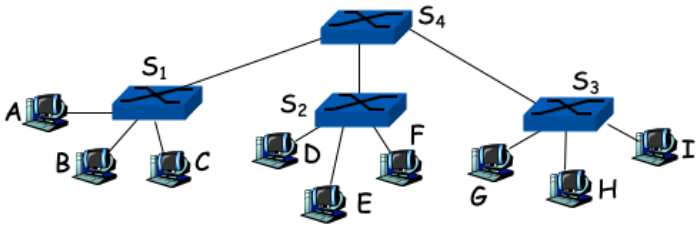
MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table (initially empty)

This slide provides an example of how the self-learning process works in a switch.

Interconnecting switches

- switches can be connected together



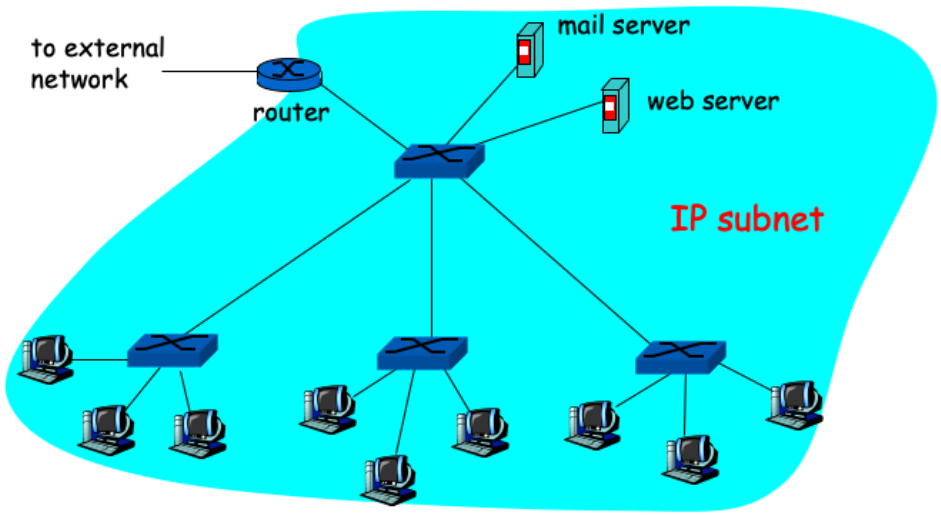
- Sending from A to F - how does S_1 know to forward frame destined to F via S_4 and S_3?
- Self learning! (works exactly the same as in single-switch case!)

ECSE 416, Lecture 17

Switches can be connected together to form a tree topology.

The self learning process works in the same way as it does for the single-switch case.

Institutional network



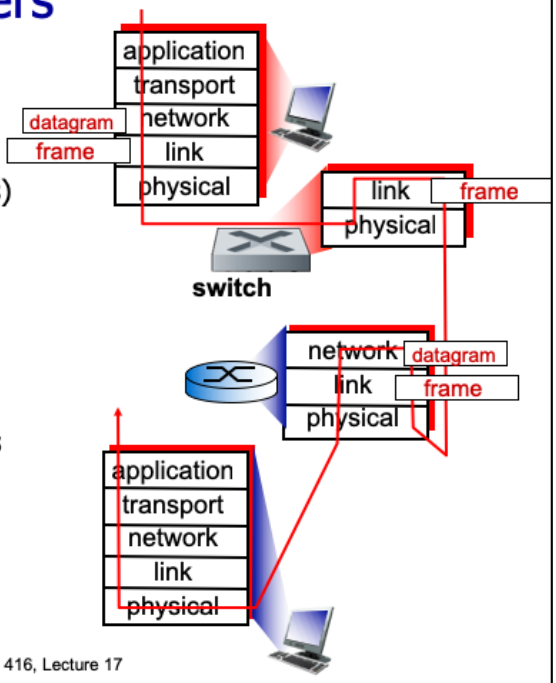
Switches vs. routers

both are store-and-forward:

- *routers:* network-layer devices (examine network-layer headers)
- *switches:* link-layer devices (examine link-layer headers)

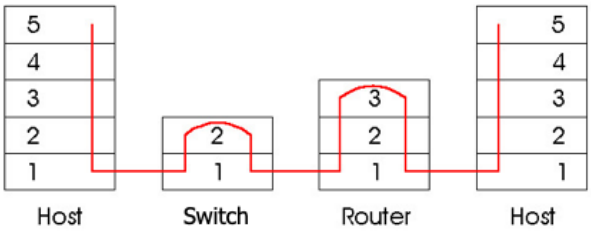
both have forwarding tables:

- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses



Switches vs. Routers

- Both store-and-forward devices
  - Routers: network layer devices (examine network layer headers)
  - Switches are link layer devices
- Routers maintain routing tables, implement routing algorithms
  - Routing topology unconstrained (can have redundancy)
- Switches maintain switch tables, implement filtering, learning algorithms
  - Switch topology must be a tree. Why?



ECSE 416, Lecture 17

Both switches and routers perform buffering. Routers are network layer (layer 3) devices; switches are link layer (layer 2) devices.

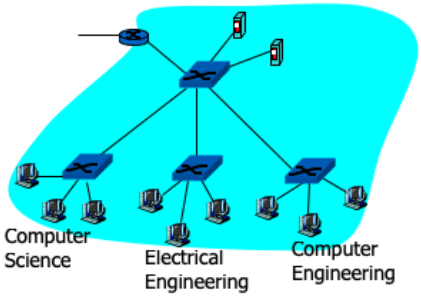
Routers have forwarding tables and coordinate to run routing algorithms that populate these tables.

The routing topology is unconstrained (although we don't want loops!). We can have redundancy, identifying back-up paths to some destinations.

A network consisting only of switches must have a tree topology. Otherwise we can experience broadcast storms. If a switch does not have an entry for a MAC address it broadcasts it on all outgoing links. In a tree topology, each unidentified address will generate a single copy on each link. In a topology with cycles the broadcasting will continue to cycle throughout the network.

Virtual LANs: motivation

What's wrong with this picture?



What happens if:

- CS user moves office to EE, but wants to connect to CS switch?
- single broadcast domain:
  - all layer-2 broadcast traffic (ARP, DHCP) crosses entire LAN (security/privacy, efficiency issues)
- each lowest level switch has only few ports in use

ECSE 416, Lecture 17

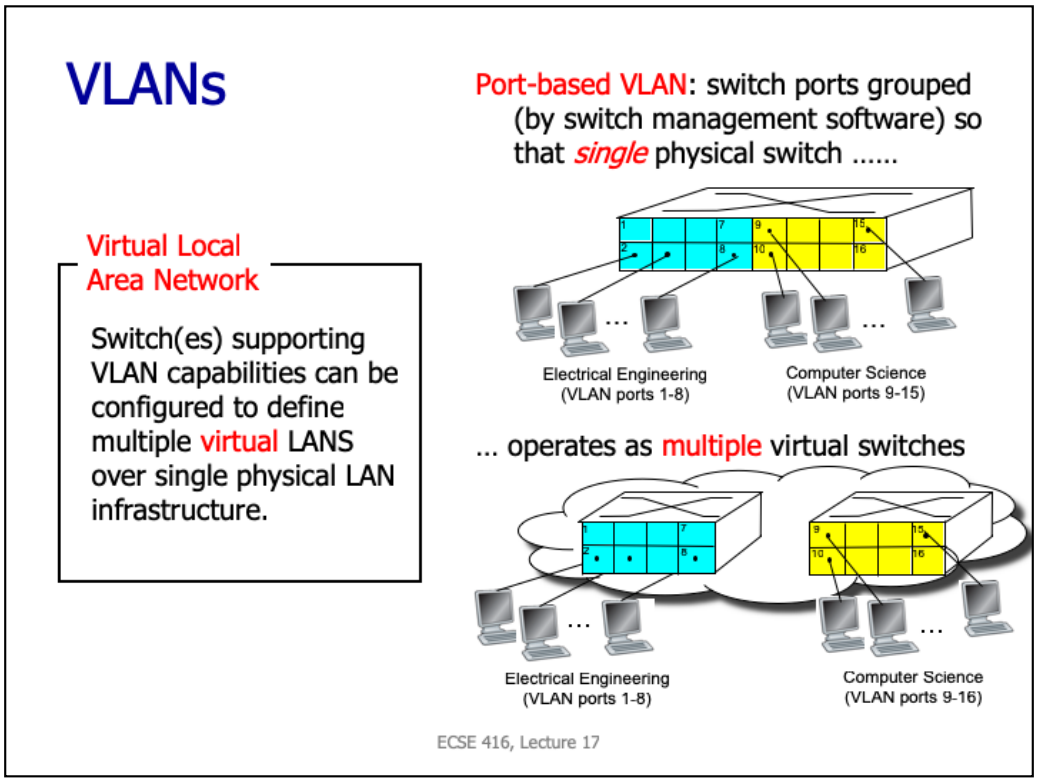
Over the past decade, networking has become increasing virtualized.

In the physical world, reconfiguration of a network can pose significant changes

In the example above, a CS user changes office, but wants to remain connected to the CS switch. There could be multiple reasons for this. If he/she has a server on the other side of the switch, then there is no need for traffic between the server and host to leave the local network. Remaining connected to the CS switch would improve security and make the transfer of data more efficient.

Given the physical nature of the network, it's very difficult to make the adjustment and provide the requested connectivity.

The other problem with the depicted topology is that the local area network is relatively large (encompassing three departments), and broadcast traffic will travel throughout this entire network.



We can improve things by constructing virtual local area networks (VLANs).

We now have a single physical switch, but define a VLAN by grouping a subset of the ports on the switch. In this way, a single physical switch operates as multiple virtual switches.

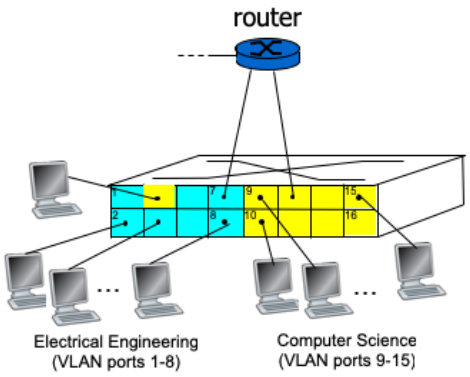
We have a single physical LAN infrastructure, but it operates logically as multiple virtual LANs.

Port-based VLAN

- *traffic isolation:* frames to/from ports 1-8 can *only* reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port

- dynamic membership: ports can be dynamically assigned among VLANs

- forwarding between VLANS: done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers



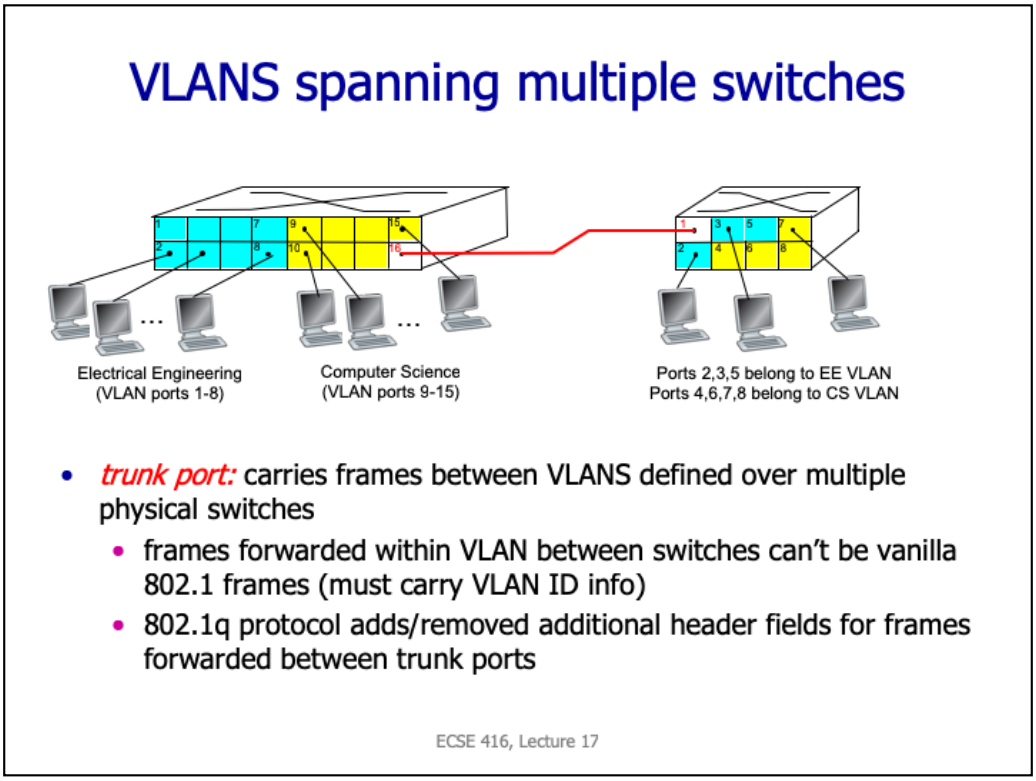
ECSE 416, Lecture 17

With this port-based VLAN approach, we can provide nice networking features relatively easily.

The gateway router can be connected to multiple ports.

We can then implement traffic isolation, so that broadcast frames within a VLAN only reach the hosts connected to the ports in the VLAN. If a frame needs to exit the VLAN, it is processed by the attached router.

We can easily perform dynamic membership changes. A physical port connected to an ECE office can now easily be changed to be logically associated with the CS VLAN.

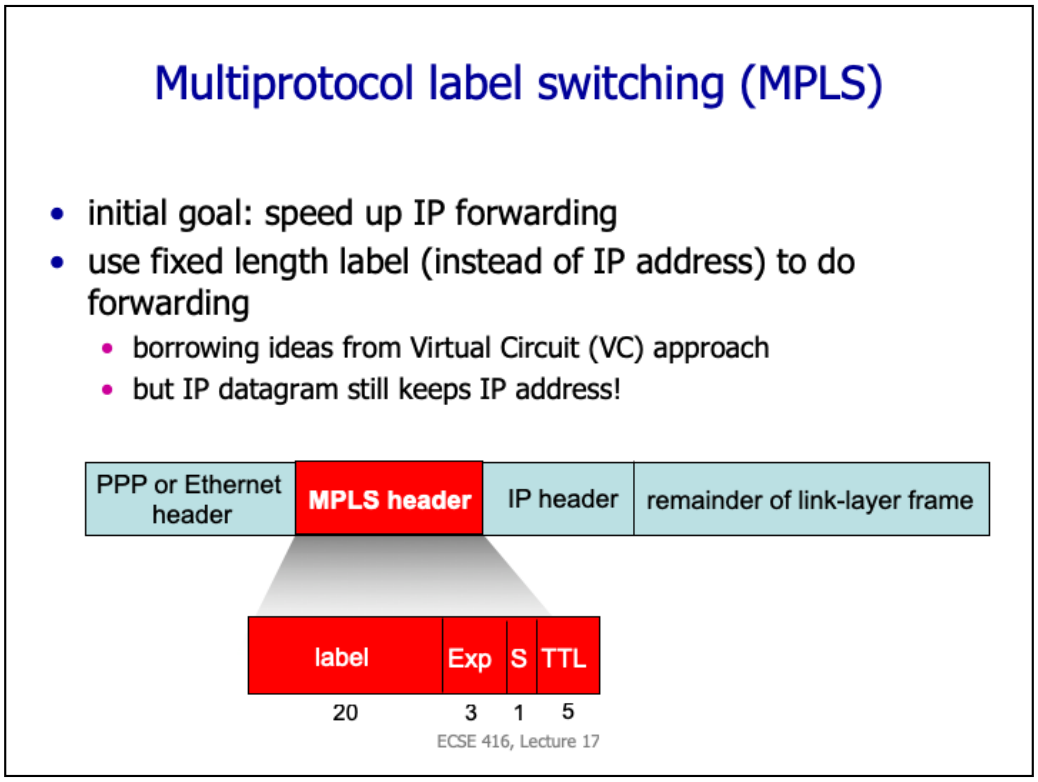


In some cases we can have a physical infrastructure that enforces the use of multiple physical switches.

We can still construct VLANs that span multiple switches. This is achieved by identifying a trunk port that carries frames from one switch to another.

Some additional information has to be inserted into the frame, because the receiving switch needs to know which VLAN the frame belongs to.

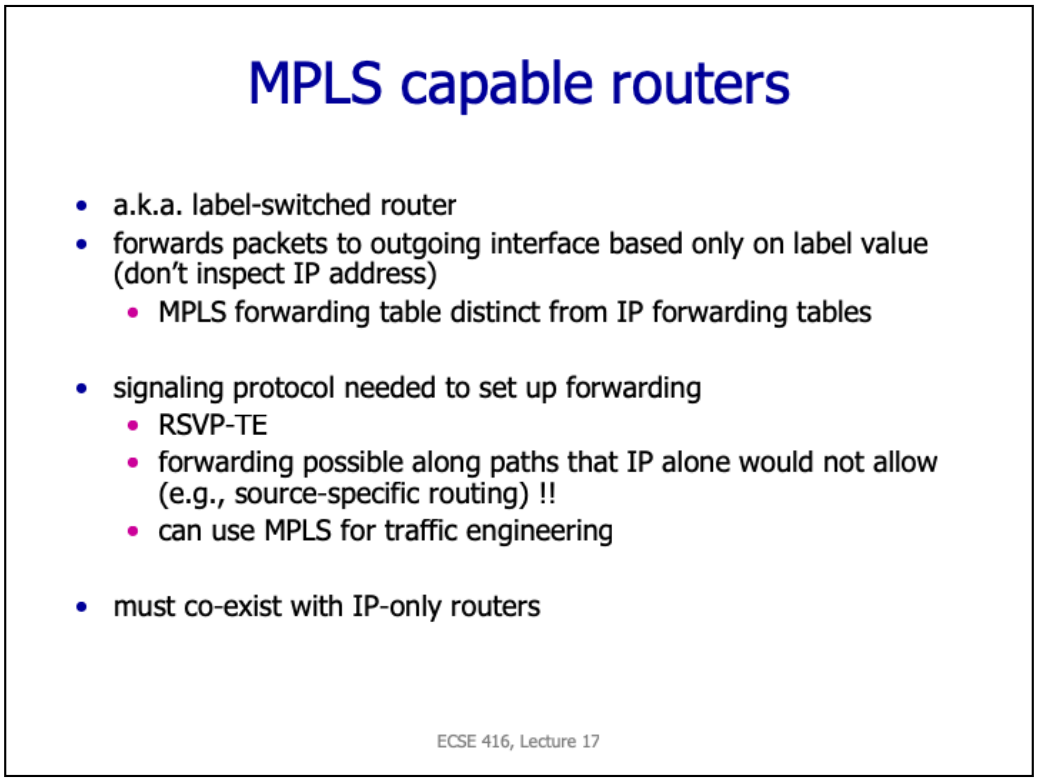
The 802.1 standards focus on protocols and architectures for local area networks and metropolitan area networks. The 802.1q protocol allows additional header fields to be inserted (including VLAN ID information) for frames that are forwarded on trunk ports.



One of the initial goals of MPLS was to speed up IP forwarding.

The key concept was to use a fixed-length label. The goal was not to replace IP addresses and the datagram-forwarding infrastructure, but to augment the infrastructure and add header information so that routers could used the fixed length labels when possible.

The MPLS header is relatively short. It consists of a label, a time-to-live, three bits that were reserved for experimental use (Exp), and an S bit to indicate the end of stacked MPLS headers (this is an advanced topic that we won't discuss in this course).



An MPLS capable (or label-switched) router can use the MPLS labels to make its forwarding decisions. There is no need for it to inspect the IP address.

The MPLS forwarding table is distinct from the IP table.

A signaling protocol is used to set up the forwarding for a specific flow.

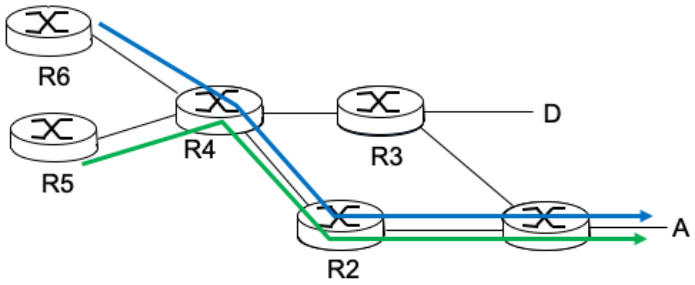
MPLS uses RSVP-TE (Resource Reservation Protocol - Traffic Engineering).

RSVP-TE allows nodes to specify desired characteristics of a flow (e.g., bandwidth, jitter) and even dictate the route traversed by the packets.

Traffic engineering involves adapting the routing and bandwidth assigned to data flows to ensure that a network doesn't become overloaded. Since MPLS allows the specification of routes and bandwidths for individual flows, it can be used to perform traffic engineering.

Not all routers support MPLS, so it must be able to work in conjunction with IP-only routers.

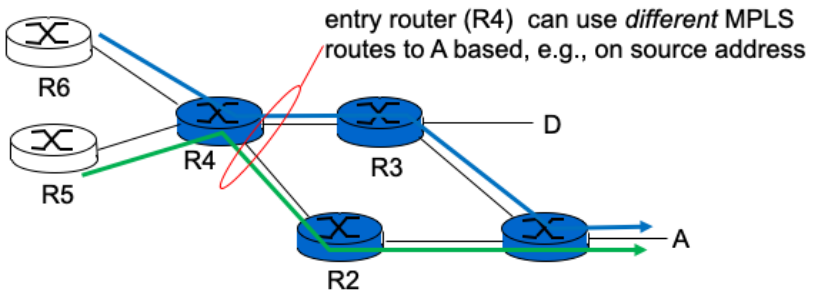
MPLS versus IP paths



- *IP routing: path to destination determined by destination address alone*

IP router

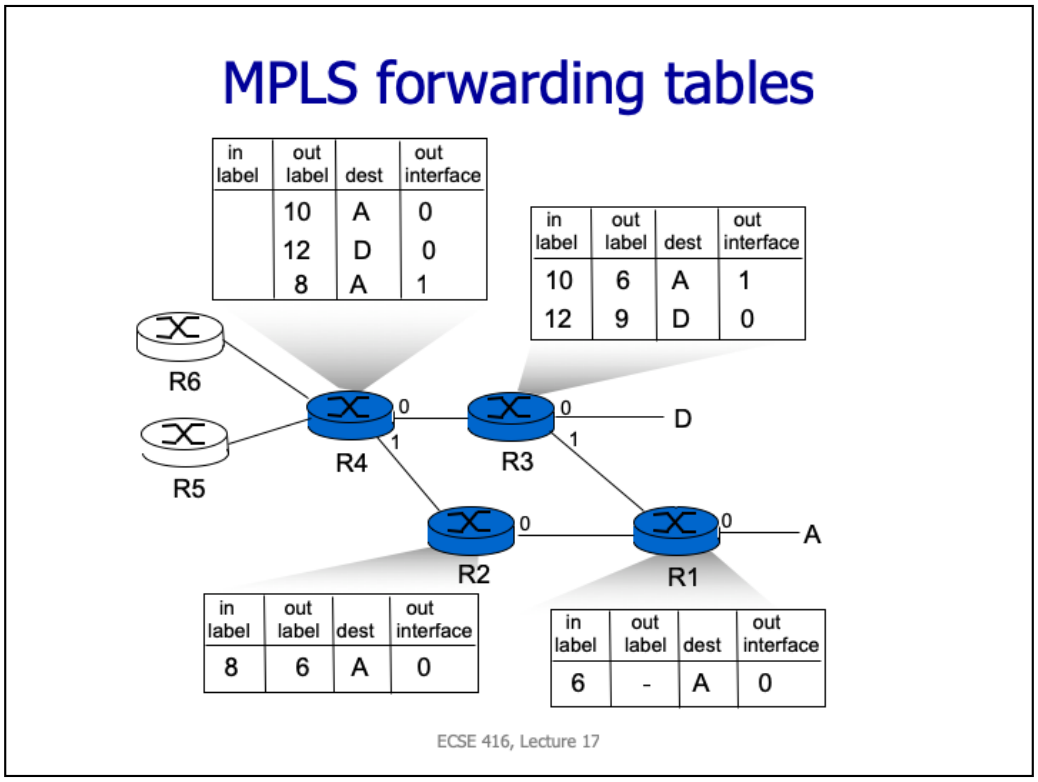
MPLS versus IP paths



- *IP routing: path to destination determined by destination address alone*
- *MPLS routing:* path to destination can be based on source *and* dest. address
  - *fast reroute:* precompute backup routes in case of link failure

IP-only router

MPLS and IP router



The MPLS forwarding tables store entries that consist of (the incoming label, the outgoing label, the destination, and the output interface).

For the same flow, there are different labels on different links. Otherwise, a flow label would need to be unique for all the routers on the path. We would need a much bigger space of possible labels to guarantee this uniqueness. By allowing the labels change on each hop, we only need local uniqueness.